GETTING IT RIGHT, UNDER CONSTRAINT: EVALUATING AND IMPROVING LLM CREATIVITY

Anonymous authors
Paper under double-blind review

ABSTRACT

One psychological account defines creativity as *constrained problem solving*, shaped by five situational factors: pressure, pleasure, opportunity, prototypes (earlier works to imitate and transform), and helpers. Current creativity benchmarks for large language models (LLMs) vary none of these factors. Their evaluation either scores correctness with the conventional route open or closes it without comparing against an unconstrained counterpart, and we find it is unstable: swapping incidental numbers alone costs models 8.9 to 15.2 points on OMEGA. We therefore build OMEGA-Constraint and Omni-MATH-Constraint, pairing each problem with versions that ban the reference method, cap length, or forbid a character, closing the conventional route while another stays open. We then add tool access as opportunity, threats as pressure, and rewards as pleasure. Whereas constraint can sharpen human creativity, LLMs lose accuracy as constraints accumulate (under all three, gpt-5.5 drops from 97.5 to 49.0 percent on OMEGA) and reapply banned methods under new names. As in humans, tools help every model, but threats and rewards of up to a million dollars move none by more than 6 points. Our proposer–composer harness, based on stage theories of creativity, supplies the remaining two factors: a small proposer (the helper) brainstorms candidates (the prototypes), which a larger composer develops into its own answer. With the best of ten proposers, both 32B composers gain in every condition, and a single 3.8B proposer lifts gpt-4.1 from 9.0 to 25.0 percent under all three constraints on OMEGA. On constrained math, our harness with a String Seed of Thought-prompted proposer beats plain and fine-tuned proposers in 17 of 28 conditions.

1 INTRODUCTION

Language models are now embedded in systems that propose hypotheses (Gottweis et al., 2026), generate research ideas (Lu et al., 2026; Si et al., 2025), invent algorithms (Novikov et al., 2025), and disprove decades-old mathematical conjectures (Alon et al., 2026; Gao, 2026). The idea-generation stage of such systems is now scored as scientific creativity (Ruan et al., 2026), yet these measures remain unsettled and unvalidated (Schapiro et al., 2026a). Accuracy-focused problem solving, by contrast, is well charted, from GSM8K (Cobbe et al., 2021) to FrontierMath (Glazer et al., 2024). Frontier systems solve olympiad and open problems at a cost of days of search (Hubert et al., 2026) or hundreds of dollars per problem (Tsoukalas et al., 2026): effective, but not economical.

Creativity has a longer history in psychology (Wallas, 1926; Guilford, 1950); beyond an initial definition as originality paired with effectiveness (Runco & Jaeger, 2012; Boden, 2004), it has recently been defined as *constrained problem solving* (Kupferberg, 2021): a solution is creative when it meets the problem's constraints effectively and economically. Creativity, on this definition, starts once the conventional route to a solution is closed and another has to be found. Kupferberg (2021) lists five situational factors that shape creative work: *pressure*, the risk and strain that push one forward; *pleasure*, the intrinsic absorption that pulls one in; *opportunity*, access to people, places, and tools to learn from, watch, and borrow from; *prototypes*, earlier works to imitate and transform; and *helpers*, the people who take part in the work itself.

Existing creativity benchmarks do not capture this definition. Some score correctness with the conventional route open, so a conventional solution and a creative one earn the same credit. Others close the conventional route but do not compare each problem with its unconstrained counterpart,

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



Figure 1: **Overview. Left:** a problem under a method ban; a frontier model alone reapplies the banned method under a new name. **Middle:** the five factors that shape creative work, each with its counterpart in our experiments (both §3.2). **Right:** a small proposer brainstorms candidate solutions and a larger composer develops them into an answer that respects the constraint (§3.3).

so a failure under the constraint cannot be distinguished from a failure at the task itself. None of them varies the situational factors (§2). Each also evaluates a single variant of every problem. Yet performance changes when we swap incidental numbers or objects while preserving the solution, and a single problem's result varies across repeated samples (§4.2).

We therefore build OMEGA-CONSTRAINT and OMNI-MATH-CONSTRAINT from two math benchmarks (Sun et al., 2026; Gao et al., 2025). Each problem is paired with versions that close the conventional route while another stays open, by banning the reference solution's method, capping the response length, or forbidding a character from the problem statement (§3.2). We measure creativity as problem solving under constraint and vary the five situational factors (Figure 1, left and middle). Constraint sharpens human creativity (Stokes, 2005; Kupferberg, 2021; Onarheim & Biskjær, 2013); we find that models under constraint lose accuracy, and more with each constraint added. We report two measures. *Compliance* measures instruction following: a model that reapplies the banned method under a new name has not complied. *Accuracy* counts a non-compliant response as wrong and measures creativity: once the model has left the conventional route, the question is whether it still reaches the answer. Tools amplify human creativity (Shneiderman, 2007). We grant opportunity as tool access: every model gains accuracy, yet banning a single library leaves some models with no answer at all. We apply pressure and pleasure as a threat and a reward, respectively, and find that they barely change accuracy.

We find that modeling the creative process in stages makes a large difference to performance. Psychology describes the creative process as a sequence (Wallas, 1926; Xie, 2025). In the variation–selection tradition, ideas arise through blind variation and are mostly mediocre; they become creative only when a later phase selects and develops the few worth keeping (Campbell, 1960; Simonton, 2023; Sawyer & Henriksen, 2024; Thompson, 2009). A language model answering in one pass collapses these stages into a single generation. Building on this, we introduce a lightweight proposer–composer harness (PCH; Figure 1, right) that gives each stage its own model and supplies the remaining two factors, helpers and prototypes: a *proposer* (the helper) brainstorms candidates without judging them, and a *composer* develops these candidates (the prototypes) into its own answer. Whether we run the proposer plain, prompt it with String Seed of Thought (SSoT) (Misaki & Akiba, 2026) for diversity, or fine-tune it, PCH raises the composer above its solo performance on every benchmark.

Our contributions are: (1) we ground LLM creativity evaluation in constraint theory, imposing constraint and varying its five situational factors on two math benchmarks; (2) we build symbolic variants of three creativity benchmarks as a robustness test and find that performance degrades on each under solution-preserving swaps (§4.2); (3) we improve creativity with PCH: given the best of ten proposers, it raises two composers on every benchmark, by up to 35% on the DRAT word-association task (Schapiro et al., 2026a) and by up to 66.0 accuracy points on OMEGA-CONSTRAINT. These results give the psychology of creativity two roles: it sets a standard that current models fall short of, and it shows how to improve them.

2 Related Work

Three kinds of benchmark test LLM creativity. The first applies psychometric tests of divergent thinking, the Divergent Association Task (DAT) and the Alternative Uses Test (Schapiro et al., 2026b), or tests associative creativity, the linking of distant concepts (Wadhwa et al., 2026). The second reframes creativity as generalization: OMEGA (Sun et al., 2026) tests exploratory, compositional, and transformative generalization after Boden's typology (Boden, 1998), scored for correctness alone. The third imposes physical constraint: MacGyver gives everyday problems in which the object properties rule out the conventional solution (Tian et al., 2024), and CreativityBench requires repurposing objects by their physical attributes (Qian et al., 2026b).

The third kind is the closest to our definition. Constraint takes two forms (Kupferberg, 2021): *physical constraint* fixes where, when, and with what materials one works; *knowledge constraint* fixes which methods and ideas one may use. MacGyver and CreativityBench apply physical constraint. Knowledge constraint appears in two of the eight tasks in CREATIVITYPRISM, which restrict the method and score correctness alongside novelty (Hou et al., 2025). Outside creativity evaluation, constrained text generation (Yao et al., 2024) restricts the form of the output and scores constraint satisfaction without checking correctness. These benchmarks do not vary the five situational factors that shape creative work, and those that impose a constraint do not compare each problem with its unconstrained counterpart, so the cost of the constraint goes unmeasured.

Our PCH differs in structure from existing methods for improving creativity. The closest to it are sampling-based inference methods, self-consistency (Wang et al., 2022) and best-of-N (Snell et al., 2024), which draw candidates from the answering model and select one. Self-Refine (Madaan et al., 2023) is also close: one model critiques and revises its own output. In PCH, by contrast, variation comes from a proposer, usually much smaller, and its candidates do not have to be correct; selection is development, the composer writing its own answer rather than choosing one. We compare against all three in Appendices U and V; further related work is discussed in Appendix A.

3 Framework

Constraint theory supplies the evaluation side of our framework, and the staged view of the creative process, the improvement side: §3.1 builds the robustness test for creativity benchmarks, §3.2 turns the definition and factors into conditions on math benchmarks, and §3.3 introduces PCH.

3.1 Solution-preserving perturbations

We begin by varying incidental objects or numbers symbolically, keeping the conventional route open, and asking whether a model still solves the problem.

Perturbations. We audit 200 problems each from MacGyver, CreativityBench, and OMEGA. On MacGyver, our hand-built rules map objects to functional categories; a replacement has the same function as the original, so every step of the reference solution still works. For each problem and each $k \in \{1, \ldots, 4\}$, we draw ten variants that pick k objects at random and replace them with random alternatives. On CreativityBench, where the solution repurposes one part of a target object, we rename k other scene objects with synonyms under the same protocol, keeping the objects used in the solution untouched, so the reference answer is unchanged. On OMEGA, we sample five numeric variants per problem under invariants preserving method and difficulty (examples in Appendix B).

Scoring. We score MacGyver and CreativityBench in two ways: by token recall against the reference solution, a deterministic metric that tracks human correctness judgments (Adlakha et al., 2024), and, following the benchmarks' own protocol, by an LLM judge that grades each answer on five-point rubrics against the same reference (prompts and rubrics in Appendix D); we score OMEGA by exact match. Beyond the mean gap, we track how each problem's accuracy varies across its own variants and across repeated samples of one prompt, since a stable mean can hide swings within single problems.

3.2 EVALUATING CREATIVITY UNDER CONSTRAINT AND THE FACTORS

Constraint goes into the problem statement, and the five situational factors vary the situation around it. Of these, "opportunity" varies access to tools, while "pressure" and "pleasure" raise the stakes through a threat for failure or a reward for success. *Helpers* and *prototypes* arrive together in PCH (§3.3). Every condition below is paired with an unconstrained baseline. The baseline asks whether a model can solve the problem, and the condition asks whether it still can once the conventional route is closed or the situational factors change.

Constraint. Creative problem solving means reaching a solution when the conventional route is closed (Kupferberg, 2021), a deliberate departure from convention (Sternberg & Lubart, 1995). The conventional route is the reference solution, and each of our three constraints closes it differently while a correct solution satisfying the constraint remains. A *method* constraint bans the reference solution's method; every banned method has a standard alternative, verified per problem (Appendix G). A *length* constraint asks for parsimony, a mark of a creative product (Kupferberg, 2021): it caps the response at 50% of the reference's word count, enough for a direct solution, since the reference is a worked derivation and the answer only a few words. A *character* constraint acts on the form of the response, a kind of constraint shown to make human writing more creative (Haught-Tromp, 2017). It forbids one character from the problem statement: its variable (the x in "solve for x"), or its most frequent character when it has no variable, and never one in the reference answer. The three constraints, alone and combined, give $2^3 = 8$ conditions, including the baseline; we call this the *constraint grid*. Length and character compliance are checked programmatically. Method compliance needs an LLM judge, since a model can reapply a banned method under a new name that a keyword check would miss. From the solution steps, the judge determines whether the banned method is used (validated against human labels, Appendix H).

Opportunity. Here the conventional route is a Python library. Models act as agents for up to six steps, each a Python tool call or a text reply. The *open* condition allows every available library; the *ban* condition forbids the reference solution's library, enforced at execution: a banned call returns an error. We evaluate the ban in two ways: *ban-eff* scores the final answer alone, so a run that calls the banned library but still answers correctly counts, and *ban-pen* marks any run with a banned call as wrong. The libraries and their verified alternatives, the ban prompt, and the models' step use are given in Appendices I and D.3.

Pressure and pleasure. A model receives pressure and pleasure only through the prompt, as a threat or a promised reward. On unconstrained problems, such prompts have been found to leave accuracy unchanged (Meincke et al., 2025); we ask whether the same holds under constraint, where a correct answer is a creative one: in humans, pressure changes creative performance, and rewards offered for creativity raise it (Byron et al., 2010; Byron & Khazanchi, 2012). We present each problem under six prompts: nothing appended (plain), a neutral sentence, a threat of consequences, and a reward at three doses, \$1, \$100, and \$1,000,000 (the sentences are given verbatim in Appendix D). However, pressure and pleasure act on humans through emotion, which a prompt cannot reproduce (§6).

3.3 PROPOSER–COMPOSER HARNESS (PCH)

Psychology divides the creative process into two stages: ideas are first generated freely, without being judged, and then the few worth keeping are selected and developed (Wallas, 1926; Campbell, 1960; Simonton, 2023; Sawyer & Henriksen, 2024; Thompson, 2009). PCH gives each stage to a different model, run in sequence. A *proposer* brainstorms candidate solutions with no obligation to be correct: blind variation. We then instruct the *composer* to treat the candidates as brainstorming material, writing its own answer from them and its own knowledge rather than copying one (formalized in Figure 2).

Development task and baselines. We develop PCH on the task of the Divergent Remote Association Test (DRAT) (Schapiro et al., 2026a): given four remote cue words (the anchors), produce ten semantically diverse nouns, each metaphorically associable with all four; the judge rates how well each noun fits the cue words, and an item is scored as the mean rating over its ten nouns. Appendix E

Roles. A proposer p supplies brainstorming material; a composer c writes the final answer.

Step 1: brainstorming material. For a problem x, the material B is one of

$$B = \begin{cases} \varnothing & \text{no proposer: composer alone} \\ W = \{w_1, \ldots, w_m\},\ w_i \sim \mathrm{Unif}(\mathcal{V}) & \text{random words from a dictionary } \mathcal{V} \\ D = \{d_1, \ldots, d_m\},\ d_i \sim p(\cdot \mid x) & \text{candidates from the proposer } (p = c\text{: self-proposal}) \end{cases}$$

Step 2: composition. The composer receives B as context and writes the answer $y = c(x, B)$.

Figure 2: Formalization of PCH.

also reports their divergence and a human validation of the judge. Two baselines run without a proposer. The first is the composer alone ($c(x, \varnothing)$ in Figure 2), with and without SSoT prompting, which asks the model to output a random string first and derive its answer from it; nothing else changes. The second, $c(x, W)$, keeps the composer prompt but replaces the candidates with random dictionary words, testing whether injected variety alone explains any gain. Against these we run ten proposers per composer (§4): nine other models, plus a second instance of the composer, $p = c$, separating the two-step structure from the use of a different model. We run each proposer in three modes: plain, prompted with SSoT, or divergent-word-tuned (fine-tuned on DRAT, next paragraph), each contributing m=20 candidates per problem.

Fine-tuning the proposer. We fine-tune the proposers to improve their brainstorming, since the composer can develop only the candidates it receives. In the divergent-word-tuned mode, we train the nine proposers with a cross-entropy objective on one shared supervised set of synthetic DRAT puzzles disjoint from the test items, and each composer on its own answers to the same puzzles (settings in Appendix J).

Transfer. Beyond DRAT, we run PCH on the constraint grid of §3.2 (OMEGA-CONSTRAINT and OMNI-MATH-CONSTRAINT) and on the perturbation sets of §3.1 (MacGyver and CreativityBench). The roles, proposers, and proposer modes are the same as on DRAT, with each proposer contributing m=5 candidates per problem (see Appendix D.5 for the prompts).

4 EXPERIMENTS

Following the setup in §4.1, our experiments are organized by the framework: §4.2 tests the robustness of three creativity benchmarks, §4.3 evaluates creativity under constraint and the situational factors, and §4.4 improves creativity with PCH.

4.1 EXPERIMENTAL SETUP

Models. Models appear in four roles: solver, composer, proposer, and judge. The solvers come from a pool of ten models: gpt-4.1 (OpenAI, 2025), gpt-5.4-mini and gpt-5.5 (Singh et al., 2025), claude-sonnet-4.6 (Anthropic, 2026), kimi-k2-thinking (Kimi Team et al., 2025), Qwen3-32B (Yang et al., 2025), DeepSeek-V4-Flash (hereafter DeepSeek-V4; Xu et al., 2026), gpt-oss-120b (Agarwal et al., 2025), Mistral-24B (Liu et al., 2026), and Phi-4-mini (Abouelenin et al., 2025); each experiment uses a subset, listed with its results. PCH adds two composers, Qwen3-32B and OLMo-2-32B (OLMo et al., 2024), nine proposers of 124M–3.8B parameters (listed in Appendix L), and the self-proposal baseline ($p = c$). The judge is gpt-oss-120b, chosen for two reasons: as an open-weight model its verdicts are reproducible, and it agrees with human verdicts more closely than even frontier models such as gpt-5.4 and gemini-2.5-pro (Verma et al., 2026). No model judges its own output, so where gpt-oss-120b is itself a solver (§4.3), Qwen3.6-35B-A3B (Qwen Team, 2026) judges in its place. On a shared set of responses, the two judges agree on 92.3%, with disagreements split evenly in both directions, so their verdicts are comparable (Appendix H).

Data and protocol. The benchmarks are those introduced above plus OMNI-MATH (Gao et al., 2025), a set of competition problems from AMC/AIME up to IMO level. Every benchmark is used as a 200-problem subset (60 for DRAT); the subsets, sources, and licenses are listed in Appendix K.

Table 1: **Existing creativity benchmarks are not robust to symbolic variation.** We vary incidental components of each problem, keeping the reference solution valid, and performance degrades. OMEGA is graded by exact match, MacGyver and CreativityBench at k=4 swaps by token recall (both in %). Orig. is the mean over the original problems and Symbolic the mean over their symbolic variants, with the arrow giving the drop from the original; σ_w is the standard deviation across one problem's variants, averaged over problems, and Worst is the average of each problem's lowest variant. A single problem swings widely across its variants, and by a similar amount across samples of one prompt (Appendix C).

	OMEGA				MacGyver				CreativityBench			
Model	Orig.	Symbolic	σ_w	Worst	Orig.	Symbolic	σ_w	Worst	Orig.	Symbolic	σ_w	Worst
gpt-4.1	40.5	↓15.2 25.3	15.2	10.5	55.5	↓3.6 51.9	6.2	42.5	21.2	↓1.3 20.0	2.8	15.7
gpt-5.4-mini	48.0	↓12.6 35.4	14.3	19.5	54.4	↓2.7 51.7	7.6	39.4	24.4	↓1.8 22.6	2.9	18.0
Qwen3-32B	29.5	↓8.9 20.6	13.9	5.8	50.5	↓3.3 47.2	7.8	34.1	21.0	↓1.2 19.7	3.4	14.7

Decoding settings and the generation budget of each run (2,048 new tokens unless stated otherwise) are given in Appendix L. Throughout, a change is called significant at $p < .05$, by an exact McNemar test when the per-problem outcome is binary (accuracy) and by a paired permutation test when it is numeric (judge ratings, response length); both tests are described in Appendix M.

4.2 ROBUSTNESS UNDER SOLUTION-PRESERVING PERTURBATIONS

Table 1 reports the audit of §3.1, with no constraint imposed. On OMEGA, replacing the numbers alone costs 8.9 to 15.2 points of accuracy; the drop holds in three independent samples, and harder arithmetic does not explain it (Appendix C). On MacGyver and CreativityBench, token recall drops by 1.2 to 3.6 points, beyond the movement of the originals across samples; under the judge the means move little, but rewording widens a problem's range beyond resampling (Appendix F). The mean understates the instability of a single problem: across its variants, accuracy swings with a standard deviation (σ_w) of up to 15.2 points, and each problem's worst variant averages less than half the accuracy on the originals. Every variant keeps the solution (§3.1), so a benchmark that scores each problem in one variant and from one sample, as all three report, cannot separate creativity from sensitivity to either. Our evaluation compares each problem with its own constrained version and tests the paired difference (§4.1), so noise from either source does not decide the result.

4.3 CONSTRAINT AND THE SITUATIONAL FACTORS

Closing the conventional route sharpens human creativity by forcing a search for another (Stokes, 2005; Kupferberg, 2021; Haught-Tromp, 2017; Acar et al., 2019); we test models across the constraint grid, and then vary the situational factors: opportunity, pressure, and pleasure.

Constraint. A response that violates any active constraint counts as wrong, so accuracy is bounded by compliance; Table 2 reports both: compliance as instruction following and accuracy over all 200 problems as creativity. The compliance judge agrees with two annotators' adjudicated labels on 87.0% of 200 responses (Appendix H). With each constraint added, models find another route less often; under all three, every model on both benchmarks falls below its baseline. Among the single constraints, only length ever ties or raises accuracy, and only slightly: gpt-5.5 stays at its baseline on OMEGA, and both gpt models gain a few points on OMNI-MATH. Even this is a budget artifact: the length constraint steers these models toward shorter answers that finish inside the 2,048-token budget, so fewer runs are truncated, and by 4,000 tokens gpt-5.4-mini's length-constrained accuracy is back below its baseline (Table 3). Every other constraint, alone or combined, costs every model accuracy, on OMEGA-CONSTRAINT the character constraint most. For gpt-5.5 and gpt-5.4-mini the reasoning tokens are hidden, so compliance can be checked only on the answer; a banned method or character used in the reasoning goes undetected. We therefore add two models with visible reasoning: kimi-k2-thinking, which shows its full reasoning trace, and claude-sonnet-4.6, which shows a summarized one. The first never satisfies the length or character constraint, and the second does so only in a minority of runs. Even the method constraint costs accuracy: on OMEGA-CONSTRAINT, claude-sonnet-4.6 complies on 170 of 200 problems yet still drops 19.5 points, and 16.5 on OMNI-MATH-CONSTRAINT, so the cost comes from the search for another route. A smaller model,

Table 2: **LLMs struggle to solve simple tasks when combined with constraints.** We show accuracy (%) across the constraint grid, the share of all 200 problems answered correctly and compliantly, with the change from that model's baseline highlighted (green ↑: rise, red ↓: drop; a tie is unmarked), the number of compliant responses of 200 in parentheses, and bold marking a significant change.

	OMEGA-CONSTRAINT				OMNI-MATH-CONSTRAINT			
Condition	gpt-5.5	gpt-5.4-mini	kimi-k2-th.	sonnet-4.6	gpt-5.5	gpt-5.4-mini	kimi-k2-th.	sonnet-4.6
baseline	97.5 (200)	91.5 (200)	68.5 (138)	95.0 (200)	62.5 (163)	36.0 (104)	2.5 (8)	45.0 (127)
length	97.5 (200)	↓4.0 87.5 (200)	↓68.5 **0.0** (0)	↓74.0 **21.0** (60)	↑3.0 65.5 (176)	↑4.5 40.5 (132)	↓2.5 0.0 (0)	↓25.0 **20.0** (53)
method	↓2.0 95.5 (196)	↓6.5 **85.0** (195)	↓26.5 **42.0** (84)	↓19.5 **75.5** (170)	↓25.5 **37.0** (98)	↓18.0 **18.0** (50)	↓1.5 1.0 (2)	↓16.5 **28.5** (81)
char	↓24.5 **73.0** (151)	↓31.0 **60.5** (138)	↓68.5 **0.0** (0)	↓91.0 **4.0** (8)	↓8.0 **54.5** (136)	↓11.0 **25.0** (80)	↓2.5 0.0 (0)	↓31.0 **14.0** (30)
length+method	↓4.0 **93.5** (193)	↓12.5 **79.0** (199)	↓68.5 **0.0** (0)	↓73.0 **22.0** (56)	↓24.0 **38.5** (106)	↓17.5 **18.5** (63)	↓2.5 0.0 (0)	↓34.0 **11.0** (29)
length+char	↓37.5 **60.0** (122)	↓32.5 **59.0** (137)	↓68.5 **0.0** (0)	↓90.0 **5.0** (14)	↓24.5 **38.0** (103)	↓18.0 **18.0** (76)	↓2.5 0.0 (0)	↓40.0 **5.0** (12)
method+char	↓31.0 **66.5** (139)	↓38.0 **53.5** (124)	↓68.5 **0.0** (0)	↓94.0 **1.0** (2)	↓32.5 **30.0** (81)	↓23.0 **13.0** (47)	↓2.5 0.0 (0)	↓32.0 **13.0** (31)
all	↓48.5 **49.0** (103)	↓41.5 **50.0** (122)	↓68.5 **0.0** (0)	↓93.5 **1.5** (6)	↓43.5 **19.0** (59)	↓21.0 **15.0** (49)	↓2.5 0.0 (0)	↓41.5 **3.5** (11)

Table 3: **Even with larger token budgets, LLMs struggle under constraint.** Accuracy (%) across the constraint grid on OMNI-MATH-CONSTRAINT, with the change from the baseline highlighted (red ↓: drop; a tie is unmarked) and the number of compliant responses of 200 in parentheses. At 16,384 tokens and beyond we repeat the baseline and the method constraint alone; kimi-k2-thinking has 57 of 200 method runs truncated at 32,768 tokens and 12 at 65,536, and judging its reasoning trace with the answer moves little (59.5 and 69.5 at baseline, 27.5 and 33.0 under the method constraint). A larger budget raises baseline accuracy but does not close the gap under constraint.

	gpt-5.4-mini				kimi-k2-thinking				
Condition	3,000	4,000	5,000	16,384	3,000	4,000	5,000	32,768	65,536
baseline	44.0 (117)	46.0 (141)	50.5 (174)	57.0 (200)	4.0 (12)	6.0 (16)	10.5 (25)	57.5 (142)	66.5 (187)
length	44.0 (165)	↓1.5 44.5 (179)	↓5.5 45.0 (190)		↓4.0 0.0 (0)	↓6.0 0.0 (0)	↓10.5 0.0 (0)		
method	↓23.0 21.0 (58)	↓22.5 23.5 (76)	↓21.5 29.0 (106)	↓24.0 33.0 (134)	↓2.5 1.5 (3)	↓2.5 3.5 (12)	↓6.0 4.5 (11)	↓31.5 26.0 (65)	↓34.5 32.0 (89)
char	↓12.0 32.0 (107)	↓11.0 35.0 (133)	↓14.0 36.5 (141)		↓4.0 0.0 (0)	↓6.0 0.0 (0)	↓10.5 0.0 (0)		
length+method	↓21.0 23.0 (78)	↓24.5 21.5 (100)	↓26.5 24.0 (117)		↓4.0 0.0 (0)	↓6.0 0.0 (0)	↓10.5 0.0 (0)		
length+char	↓22.0 22.0 (83)	↓23.0 23.0 (94)	↓30.0 20.5 (113)		↓4.0 0.0 (0)	↓6.0 0.0 (0)	↓10.5 0.0 (0)		
method+char	↓28.0 16.0 (61)	↓31.0 15.0 (74)	↓34.0 16.5 (82)		↓4.0 0.0 (0)	↓6.0 0.0 (0)	↓10.5 0.0 (0)		
all	↓31.5 12.5 (58)	↓36.5 9.5 (62)	↓34.5 16.0 (79)		↓4.0 0.0 (0)	↓6.0 0.0 (0)	↓10.5 0.0 (0)		

Qwen3-32B, shows the same pattern: collapse under the length and character constraints, and an accuracy drop under the method constraint as well (Appendix N).

Ruling out truncation. We run two checks on OMNI-MATH-CONSTRAINT with gpt-5.4-mini and kimi-k2-thinking, the two models most often truncated at 2,048 tokens. The first repeats the constraint grid at budgets of 3,000, 4,000, and 5,000 tokens (Table 3). Truncation decreases while the gap holds: under all three constraints, gpt-5.4-mini stays 31.5, 36.5, and 34.5 points below its baseline, and kimi-k2-thinking stays at zero while its baseline rises from 4.0 to 10.5. The second repeats the baseline and the method constraint, the one constraint a larger budget can help, at budgets of 16,384 to 65,536 tokens. At 16,384 tokens every gpt-5.4-mini run completes and the gap holds at 24.0 points; kimi-k2-thinking, with 12 of 200 runs still truncated at 65,536 tokens, shows gaps of 31.5 and 34.5 points at its two budgets. Method compliance also varies by subject, from 82% on combinatorics to 52% on number theory for gpt-5.4-mini and from 69% to 21% for kimi-k2-thinking at 32,768 tokens; doubling its budget keeps the subjects in the same order (Appendix O).

Opportunity. Since OMEGA is near saturation without tools, we run this condition on the unconstrained OMNI-MATH problems. Tool access amplifies every model: an unrestricted Python tool adds 26.0 points for claude-sonnet-4.6, 22.0 for gpt-5.4-mini, 17.5 for kimi-k2-thinking, and 14.5 for gpt-5.5, all four significant (Table 4). Banning a single library then separates them: against the open condition, gpt-5.4-mini drops the most, 8.0 points under *ban-pen* and 3.5 under *ban-eff*; claude-sonnet-4.6 drops 1.0 under *ban-pen* and nothing under *ban-eff*; gpt-5.5 is unchanged under either scoring. Only kimi-k2-thinking gains, by 1.0 under *ban-pen* and 1.5 under *ban-eff*, but more than half of its runs end with no final answer under either condition, so the ban rarely comes into play and the gain is noise around a low floor. Opportunity raises accuracy for every model, but recovering from its withdrawal is itself a capability, and not every model has it: gpt-5.4-mini's drop under *ban-pen* is significant.

Pressure and pleasure. The appended sentences barely change accuracy on OMEGA (Figure 3): whether on the baseline problems or under constraint, no sentence moves any model by more than 6

Table 4: **Every model gains from the tool, and the ban significantly costs gpt-5.4-mini alone.** We show accuracy (%) on OMNI-MATH with no tool, with every available library (*open*), and with one library banned, scored two ways: *ban-eff* counts the final answer alone and *ban-pen* any banned call as wrong. #Viols. is the number of problems with a banned call and Empty the number with no final answer. Arrows mark the change of *open* against no tool and of the ban columns against *open* (green ↑: rise, red ↓: drop; a tie is unmarked), with bold marking a significant change.

Model	No tool	Open	ban-eff	ban-pen	#Viols.	Empty (open)	Empty (ban)
gpt-5.4-mini	36.0	↑22.0 **58.0**	↓3.5 54.5	↓8.0 **50.0**	20	36	46
gpt-5.5	62.5	↑14.5 **77.0**	77.0	77.0	0	3	0
claude-sonnet-4.6	45.0	↑26.0 **71.0**	71.0	↓1.0 70.0	2	18	24
kimi-k2-thinking	2.5	↑17.5 **20.0**	↑1.5 21.5	↑1.0 21.0	3	106	108

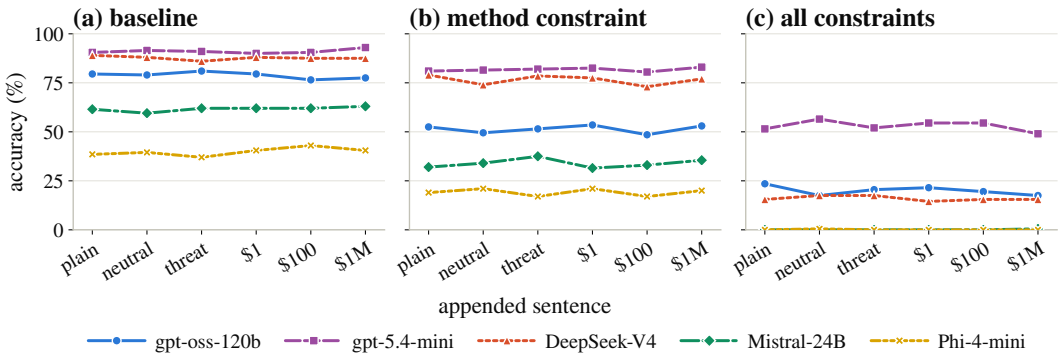


Figure 3: **Threats and rewards barely move accuracy.** We show accuracy on OMEGA for each appended sentence, on (a) the baseline problems, (b) under the method constraint, and (c) under all three constraints. One change from the plain prompt is significant: DeepSeek-V4 with the $100 reward under the method constraint.

points from the plain prompt, and only one change is significant (DeepSeek-V4, -6.0 under the $100 reward with the method constraint). Length does respond, but mostly for one model (Appendix P.1): Mistral-24B writes more with every appended sentence and in dose order, up to $+173$ words at $1,000,000 on the baseline problems and $+226$ under the method constraint, while the other models move by a few dozen words at most. The same holds on OMNI-MATH (Appendix P.2). Pressure and pleasure, which change creative performance in humans, have almost no effect on the models' creativity.

4.4 DISTRIBUTING THE CREATIVE PROCESS

Development task. On DRAT (Table 5; Appendix Q works through one item), with OLMo-2-32B as composer, every divergent-word-tuned proposer raises the judge rating significantly, from 2.34 for the composer alone to between 2.82 and 3.17 on a 0–4 scale, and the largest gain comes from a 135M-parameter proposer. Plain and SSoT proposers can help without any training, significantly in both modes for Qwen3-1.7B with OLMo-2-32B ($+0.39$, $+0.30$) and GPT-2 genre with Qwen3-32B ($+0.34$, $+0.22$); tuning extends the gain to every separate proposer. The gain needs real candidates: random words in their place lower the rating by 0.27 for OLMo-2-32B (significant) and 0.04 for Qwen3-32B, and SSoT prompting of the composer alone lowers it by 0.14 and 0.09. Self-proposal helps significantly only in the divergent-word-tuned mode and stays below the best separate proposer. Qwen3-32B gains less from the separate proposers than OLMo-2-32B does, by $+0.14$ to $+0.38$ in the tuned mode against $+0.48$ to $+0.83$, and its self-proposal lowers its rating in every mode. The gain requires a second model: Self-Refine (Madaan et al., 2023), in which the composer critiques and revises its own answer, gives -0.05 for OLMo-2-32B ($p = .63$) and $+0.27$ for Qwen3-32B ($p = .017$), against $+0.83$ and $+0.38$ for PCH (Appendix U).

Transfer. Table 6 reports the highest accuracy the composer reaches with any of the ten plain or SSoT proposers in each condition, an upper bound on proposer choice; the divergent-word-tuned

Table 5: **Boosting creative problem solving with a divergent-word-tuned proposer.** We show the judge rating on DRAT (0–4, mean over 60 items) for each composer with each proposer, run plain, with SSoT, or after divergent-word tuning (div.-tuned). *alone*: the composer with no proposer; *itself*: the composer as its own proposer; th.: thinking mode; r.: the reasoning variant. Cells give the rating and its change over *alone* (green ↑: rise, red ↓: drop; a tie is unmarked), with bold marking a significant change. Divergent-word tuning helps most, and a 135M-parameter proposer leads.

Composer	Mode	alone	SmolLM 135M	GPT-2	GPT-2 genre	OLMo-2 1B	Qwen3 1.7B	SmolLM3 3B	SmolLM3 3B th.	Phi-4 mini	Phi-4 mini-r.	itself
OLMo-2-32B	plain	2.34	↑0.14 2.48	↑0.06 2.40	↑0.13 2.47	↓0.07 2.27	↑0.39 **2.73**	↑0.26 **2.60**	↑0.29 **2.63**	↑0.08 2.42	↑0.05 2.39	2.34
	SSoT		↑0.18 2.52	↑0.16 2.50	↑0.14 2.48	↓0.07 2.27	↑0.30 **2.64**	↓0.04 2.30	↑0.01 2.35	↓0.25 **2.09**	↑0.07 2.41	↑0.16 2.50
	div.-tuned		↑0.83 **3.17**	↑0.53 **2.87**	↑0.56 **2.90**	↑0.48 **2.82**	↑0.55 **2.89**	↑0.54 **2.88**	↑0.66 **3.00**	↑0.52 **2.86**	↑0.62 **2.96**	↑0.52 **2.86**
Qwen3-32B	plain	2.10	↑0.10 2.20	↑0.14 2.24	↑0.34 **2.44**	↓0.11 1.99	↑0.28 **2.38**	↑0.19 2.29	↑0.17 2.27	↑0.16 2.26	↑0.19 **2.29**	↓0.09 2.01
	SSoT		↑0.17 2.27	↑0.19 2.29	↑0.22 **2.32**	↓0.08 2.02	↑0.26 **2.36**	↓0.03 2.07	↓0.01 2.09	↓0.15 1.95	↑0.04 2.14	↓0.16 1.94
	div.-tuned		↑0.38 **2.48**	↑0.28 **2.38**	↑0.14 2.24	↑0.37 **2.47**	↑0.35 **2.45**	↑0.34 **2.44**	↑0.37 **2.47**	↑0.26 **2.36**	↑0.30 **2.40**	↓0.09 2.01

Table 6: **Improved creativity transfers to math-constrained problem solving.** We show composer accuracy (%) alone and with PCH on OMEGA-CONSTRAINT and OMNI-MATH-CONSTRAINT. *alone*: the composer with no proposer; each mode cell (plain, SSoT) gives the composer's accuracy with the best of the ten proposers and its change over *alone* (green ↑: rise), with bold marking a significant gain. The composer gains in every cell; divergent-word-tuned proposers are in Table 23.

	OMEGA						OMNI-MATH					
	OLMo-2-32B			Qwen3-32B			OLMo-2-32B			Qwen3-32B		
Condition	alone	plain	SSoT	alone	plain	SSoT	alone	plain	SSoT	alone	plain	SSoT
baseline	8.0	↑50.0 **58.0**	↑51.0 **59.0**	70.0	↑15.0 **85.0**	↑13.5 **83.5**	4.5	↑4.5 9.0	↑3.0 7.5	21.5	↑5.0 26.5	↑3.0 24.5
length	8.0	↑64.5 **72.5**	↑66.0 **74.0**	9.0	↑13.0 **22.0**	↑19.0 **28.0**	4.0	↑10.5 **14.5**	↑11.5 **15.5**	2.0	↑7.5 **9.5**	↑8.5 **10.5**
method	8.0	↑45.0 **53.0**	↑46.5 **54.5**	50.0	↑9.0 **59.0**	↑8.5 **58.5**	4.0	↑3.5 **7.5**	↑3.5 7.5	12.0	↑2.0 14.0	↑4.5 16.5
char	6.0	↑53.0 **59.0**	↑54.5 **60.5**	0.5	↑3.0 **3.5**	↑3.5 **4.0**	4.5	↑3.0 7.5	↑4.5 **9.0**	0.0	↑0.5 0.5	↑0.5 0.5
length+method	8.0	↑59.0 **67.0**	↑59.5 **67.5**	8.0	↑2.5 10.5	↑1.5 9.5	2.5	↑11.5 **14.0**	↑9.5 **12.0**	2.0	↑5.0 **7.0**	↑6.0 **8.0**
length+char	6.5	↑60.5 **67.0**	↑64.5 **71.0**	3.5	↑19.0 **22.5**	↑13.5 **17.0**	1.5	↑13.5 **15.0**	↑14.5 **16.0**	0.0	↑1.5 1.5	↑1.0 1.0
method+char	5.5	↑45.5 **51.0**	↑49.0 **54.5**	0.0	↑0.5 0.5	↑1.0 1.0	3.5	↑2.5 6.0	↑2.5 6.0	0.0	↑0.5 0.5	↑0.5 0.5
all	4.5	↑54.5 **59.0**	↑58.5 **63.0**	1.5	↑14.5 **16.0**	↑9.0 **10.5**	1.5	↑11.0 **12.5**	↑12.5 **14.0**	0.0	↑1.0 1.0	↑0.5 0.5

proposers are in Appendix R. The composer gains in every cell, significantly in 42 of the 64, and under all three constraints OLMo-2-32B rises from 4.5 to 63.0 on OMEGA and from 1.5 to 14.0 on OMNI-MATH. Extra sampling from the composer does not produce the gain: for OLMo-2-32B, twenty samples aggregated by self-consistency move OMEGA accuracy under all three constraints from 4.5 to 5.5, and pass@20 reaches 17.5, against 21.7 for PCH averaged over the ten plain proposers and 63.0 with the best SSoT proposer (Appendix V). The composer develops the candidates rather than selecting one: under constraint, its accuracy usually exceeds what the best selection among its candidates could reach, and always exceeds its accuracy alone (Table 27). A closed-weight composer gains too: with Phi-4-mini-reasoning candidates, gpt-4.1 rises on OMEGA from 83.0 to 88.5 unconstrained, from 56.0 to 64.5 under the method constraint, and from 9.0 to 25.0 under all three, all significant, and by at most 2.0 points on the same cells of OMNI-MATH, not significant. On MacGyver and CreativityBench, with the judge rating rescaled to 0–100, the best gains reach 11.2 and 13.5 points (Appendix S). Appendix T adds a worked example, each proposer's accuracy on its own, the winning proposer per cell, and the composer's mean accuracy across the ten proposers.

5 CONCLUSION

We evaluated creativity in language models as one psychological account defines it in humans: problem solving under constraint in a situation shaped by opportunity, pressure, and pleasure. Where constraint sharpens human creativity, models lose accuracy with each constraint added: banned methods return under new names, a withdrawn library leaves some models without an answer, while rewards and threats barely move accuracy. To improve creativity, we built PCH on stage theories of creativity, which raises the composer above what it achieves alone on every benchmark. Creativity in today's language models therefore has to be built as a process; a prompt alone does not elicit it.

6 LIMITATIONS

Pressure and pleasure reach a model only through the prompt, as a threat and a reward. In humans they work through emotion, which serves two functions that LLMs lack (Goldenberg & Gross, 2026). First, emotion shapes how a situation is interpreted, and this appraisal is context dependent, the same situation generating fear in one person and excitement in another; a model's internal representations of emotional relevance are consistent rather than flexible in this way. Second, emotion reorganizes how a person responds, narrowing attention, accelerating decisions, and driving action; each forward pass of an LLM is computationally fixed, so an appended threat or reward shifts the probability of the next token but does not reorganize the processing that produces it. Alignment may also play a part: post-training has been found to reduce the activation of high-arousal emotion representations (Sofroniew et al., 2026), and learning from human feedback has been linked to insensitivity to the consequences attached to a task (Ruiz & Cardona-Betancourt, 2026).

A difference between humans and models also appears under the character and length constraints. Sub-word tokenization makes avoiding a character harder for a model than for a human, since a model generates tokens rather than characters. The demand is the one made of a human writer under the same constraint, to rename the variable or choose words without the letter and still reach the answer, far below the olympiad problems that recent work reports models solving (Hubert et al., 2026; Huang & Yang, 2025). The length limit is set at half the reference length, so complying asks for a shorter solution rather than an exact count. The method constraint needs no counting of characters or words, and its gap holds when the budget is raised (Table 3). We therefore read a failure under these constraints as a difference between humans and models facing the same demand rather than as a limit of our evaluation.

ETHICS STATEMENT

This work evaluates language models on publicly released benchmarks. Every claim about human creativity is taken from the published psychological literature, and no personal data were collected. Datasets are used under their licenses (Appendix K). Human annotations are limited to labels on model outputs, used to validate the two LLM judges (Appendices E and H).

REPRODUCIBILITY STATEMENT

Every element needed to rerun the experiments is in the paper or its appendix. All prompts are reproduced verbatim in Appendix D: solver prompts, constraint sentences, tool instructions, threat and reward sentences, PCH prompts, and judge prompts with their rubrics. The construction of the perturbation variants is described in Appendix B, the banned methods and their alternatives are listed in Appendix G, and the library ban, including the libraries, their alternatives, and the step budget, in Appendix I. The datasets, subsets, and licenses are listed in Appendix K. The model roster, decoding settings, and generation budgets are given in Appendix L, and the fine-tuning data and hyperparameters in Appendix J; the compute used and the software versions are in Appendix W. The significance tests for every experiment are specified in Appendix M, and the human validation of the two LLM judges in Appendices E and H. Code, the augmented benchmark versions, and the synthetic DRAT training puzzles will be released upon acceptance.

AI USE STATEMENT

We used generative AI tools to check the grammar and spelling of the manuscript and as an aid in debugging code written by the authors; no task with required disclosure under the ICLR policy involved generative AI. We have reviewed all AI-assisted work: every suggested edit was accepted or rejected by the authors, and suggested code fixes were applied by the authors after verification. We take responsibility for the final content of this work.

REFERENCES

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, et al. Phi-4-mini technical

report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025.

Oguz A Acar, Murat Tarakci, and Daan Van Knippenberg. Creativity and innovation under constraints: A cross-disciplinary integrative review. *Journal of management*, 45(1):96–121, 2019.

Vaibhav Adlakha, Parishad BehnamGhader, Xing Han Lu, Nicholas Meade, and Siva Reddy. Evaluating correctness and faithfulness of instruction-following models for question answering. *Transactions of the Association for Computational Linguistics*, 12:681–699, 2024.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Saeid Alavi Naeini, Raeid Saqur, Mozhgan Saeidi, John Giorgi, and Babak Taati. Large language models are fixated by red herrings: Exploring creative problem solving and einstellung effect using the only connect wall dataset. *Advances in Neural Information Processing Systems*, 36: 5631–5652, 2023.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. Smollm2: When smol goes big–data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Noga Alon, Thomas F Bloom, W Timothy Gowers, Daniel Litt, Will Sawin, Arul Shankar, Jacob Tsimerman, Victor Wang, and Melanie Matchett Wood. Remarks on the disproof of the unit distance conjecture. *arXiv preprint arXiv:2605.20695*, 2026.

Anthropic. Claude sonnet 4.6 system card. `https://www-cdn.anthropic.com/bbd8ef16d70b7a1665f14f306ee88b53f686aa75/Claude%20Sonnet%204.6%20System%20Card.pdf`, 2026.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, et al. Smollm3: smol, multilingual, long-context reasoner. *Hugging Face Blog*, 2025.

Margaret A Boden. Creativity and artificial intelligence. *Artificial intelligence*, 103(1-2):347–356, 1998.

Margaret A Boden. *The creative mind: Myths and mechanisms*. Routledge, 2004.

Kris Byron and Shalini Khazanchi. Rewards and creative performance: a meta-analytic test of theoretically derived hypotheses. *Psychological bulletin*, 138(4):809, 2012.

Kristin Byron, Shalini Khazanchi, and Deborah Nazarian. The relationship between stressors and creativity: a meta-analysis examining competing theoretical models. *Journal of Applied Psychology*, 95(1):201, 2010.

Donald T Campbell. Blind variation and selective retentions in creative thought as in other knowledge processes. *Psychological review*, 67(6):380, 1960.

Florian Carichon, Soumya Sharma, Meaghan Girard, Romain Rampa, and Golnoosh Farnadi. Ideafix: Evaluation framework for creative defixation prompting in llms. *arXiv preprint arXiv:2606.00875*, 2026.

Yanran Chen, Hannes Gröner, Sina Zarrieß, and Steffen Eger. Evaluating diversity in automatic poetry generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 19671–19692, 2024.

John Joon Young Chung, Vishakh Padmakumar, Melissa Roemmele, Yuqian Sun, and Max Kreminski. Modifying large language model post-training for diverse creative writing. *arXiv preprint arXiv:2503.17126*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

Ronald Aylmer Fisher. *The design of experiments*. Oliver and Boyd, Edinburgh, 8 edition, 1966.

Giorgio Franceschelli and Mirco Musolesi. On the creativity of large language models. *AI & society*, 40(5):3785–3795, 2025.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Zhengyang Tang, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 100540–100569, 2025.

Shuhong Gao. Counterexamples to the jacobian conjecture in dimensions greater than two. *arXiv preprint arXiv:2608.00222*, 2026.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, et al. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872*, 2024.

Amit Goldenberg and James J Gross. Large language models do not have emotions. *Nature Human Behaviour*, pp. 1–2, 2026.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, et al. Accelerating scientific discovery with co-scientist. *Nature*, pp. 1–3, 2026.

Joy Paul Guilford. Creativity. *The American Psychologist*, 5(9):444–454, 1950.

Catrinel Haught-Tromp. The green eggs and ham hypothesis: How constraints facilitate creativity. *Psychology of Aesthetics, Creativity, and the Arts*, 11(1):10, 2017.

Zhaoyi Joey Hou, Bowei Alvin Zhang, Yining Lu, Bhiman Kumar Baghel, Anneliese Brei, Ximing Lu, Meng Jiang, Faeze Brahman, Snigdha Chaturvedi, Haw-Shiuan Chang, et al. Creativityprism: A cross-domain evaluation framework for large language model creativity. *arXiv preprint arXiv:2510.20091*, 2025.

Zhaoyi Joey Hou, Zhuowei Chen, Mengxue Zhang, and Xiang Lorraine Li. Rethinking Post-training Diversity Collapse: Is Diversity-preserving Post-training Enough? In *ICML 2026 Workshop on Human-AI Co-Creativity*, 2026. Non-archival workshop paper. `https://genaicreativity.org/icml2026/files/54/54_paper.pdf` (accessed 23 August 2026).

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. *arXiv preprint arXiv:2106.09685*, 2021.

Yichen Huang and Lin F Yang. Winning gold at imo 2025 with a model-agnostic verification-and-refinement pipeline. *arXiv preprint arXiv:2507.15855*, 2025.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613, 2026.

Mete Ismayilzada, Simone A Luchini, Abdulkadir Gokce, Badr AlKhamissi, Antoine Bosselut, Antonio Laverghetta Jr, Lonneke van der Plas, and Roger E Beaty. Large language models align with the human brain during creative thinking. *arXiv preprint arXiv:2604.03480*, 2026.

Liwei Jiang, Yuanjun Chai, Margaret Li, Mickel Liu, Raymond Fok, Nouha Dziri, Yulia Tsvetkov, Maarten Sap, and Yejin Choi. Artificial hivemind: The open-ended homogeneity of language models (and beyond). *Advances in Neural Information Processing Systems*, 38, 2026.

Constantinos Karouzos, Xingwei Tan, and Nikolaos Aletras. Where does output diversity collapse in post-training? *arXiv preprint arXiv:2604.16027*, 2026.

Kimi Team, Yifan Bai, Yiping Bao, Y Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of rlhf on llm generalisation and diversity. In *International Conference on Learning Representations*, volume 2024, pp. 20620–20653, 2024.

Vanya Bannihatti Kumar, Anish Narang, Mayank Anand, Akhil Eppa, and Divyanshu Goyal. JudgeBreak: Is Taste Hidden in the Layers? Stress-Testing Reward Models and LLM Judges with Representation-Based Subjective Evaluation. In *ICML 2026 Workshop on Human-AI Co-Creativity*, 2026. Non-archival workshop paper. `https://genaicreativity.org/icml2026/files/51/51_paper.pdf` (accessed 23 August 2026).

Maria Kunilovskaya, Gagan Bhatia, Lisa Sophie Albertelli, Yanran Chen, Christian Greisinger, Lotta Kiefer, Christoph Leiter, Subhadeep Roy, Tewodros Achamaleh, Muhammad Arslan Manzoor, et al. Who annotates in nlp? a large-scale assessment of human annotation reporting between 2018 and 2025. *arXiv preprint arXiv:2606.02255*, 2026.

Feiwel Kupferberg. *Constraints and creativity: In search of creativity science*. Cambridge University Press, 2021.

Jack Lanchantin, Angelica Chen, Shehzaad Dhuliawala, Ping Yu, Jason Weston, Sainbayar Sukhbaatar, and Ilia Kulikov. Diverse preference optimization. *arXiv preprint arXiv:2501.18101*, 2025.

Jun Li. We built a salon, we called it alignment: Aesthetic reward and the latent avant-garde in generative models. In *ICML 2026 Workshop on Human-AI Co-Creativity*, 2026. URL `https://genaicreativity.org/icml2026/files/62/62_paper.pdf`.

Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. Jointly reinforcing diversity and quality in language model generations. *arXiv preprint arXiv:2509.02534*, 2025.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of ai research. *Nature*, 651(8107):914–919, 2026.

Pengzhao Lyu, Yeun Joon Kim, Hanlin Xiao, and Yingyue Luna Luan. Why large language models and humans converge and diverge in evaluating creativity. *arXiv preprint arXiv:2607.22218*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pp. 50–60, 1947.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

Pronita Mehrotra, Aishni Parab, and Sumit Gulwani. Enhancing creativity in large language models through associative thinking strategies. *arXiv preprint arXiv:2405.06715*, 2024.

Lennart Meincke, Ethan Mollick, Lilach Mollick, and Dan Shapiro. Prompting science report 3: I'll pay you or i'll kill you–but will you care? *arXiv preprint arXiv:2508.00614*, 2025.

Kou Misaki and Takuya Akiba. String seed of thought: Prompting llms for distribution-faithful and diverse generation. In *International Conference on Learning Representations*, volume 2026, pp. 89078–89136, 2026.

C Page Moreau and Darren W Dahl. Designing the solution: The impact of constraints on consumers' creativity. *Journal of Consumer research*, 32(1):13–22, 2005.

Vaishnavh Nagarajan, Chen Henry Wu, Charles Ding, and Aditi Raghunathan. Roll the dice & look before you leap: Going beyond the creative limits of next-token prediction. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*. PMLR, 2025.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Balder Onarheim and Michael Mose Biskjær. An introduction to 'creativity constraints'. In *24th ISPIM Innovation Conference: Innovating in Global Markets: Challenges for Sustainable Growth*. Lappeenranta University of Technology Press, 2013.

OpenAI. Introducing gpt-4.1 in the api. `https://openai.com/index/gpt-4-1/`, 2025.

Peter Organisciak, Selcuk Acar, Denis Dumas, and Kelly Berthiaume. Beyond semantic distance: Automated scoring of divergent thinking greatly improves with large language models. *Thinking Skills and Creativity*, 49:101356, 2023.

William Orwig and Roger E Beaty. How llms see creativity: Zero-shot scoring of visual creativity with interpretable reasoning. *arXiv preprint arXiv:2606.29672*, 2026.

Max Peeperkorn, Tom Kouwenhoven, Dan Brown, and Anna Jordanous. Is temperature the creativity parameter of large language models? *arXiv preprint arXiv:2405.00492*, 2024.

pranavpsv. gpt2-genre-story-generator. Hugging Face model repository, 2021. URL `https://huggingface.co/pranavpsv/gpt2-genre-story-generator`. GPT-2 fine-tuned for genre-based story generation. Accessed: 2026-08-23.

Cheng Qian, Peixuan Han, Qinyu Luo, Bingxiang He, Xiusi Chen, Yuji Zhang, Hongyi Du, Jiarui Yao, Xiaocheng Yang, Denghui Zhang, et al. Escapebench: Towards advancing creative intelligence of language model agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 798–820, 2025.

Cheng Qian, Hyeonjeong Ha, Jiayu Liu, Jeonghwan Kim, Emre Can Acikgoz, Bingxuan Li, Kunlun Zhu, Jiateng Liu, Aditi Tiwari, Zhenhailong Wang, et al. Advancing creative physical intelligence in large multimodal models. *arXiv preprint arXiv:2605.26396*, 2026a.

Cheng Qian, Hyeonjeong Ha, Jiayu Liu, Jeonghwan Kim, Jiateng Liu, Bingxuan Li, Aditi Tiwari, Dwip Dalal, Zhenhailong Wang, Xiusi Chen, et al. Creativitybench: Evaluating agent creative reasoning via affordance-based tool repurposing. *arXiv preprint arXiv:2605.02910*, 2026b.

Ziliang Qiu and Renfen Hu. Deep associations, high creativity: A simple yet effective metric for evaluating large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10870–10883, 2025.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Kai Ruan, Xuan Wang, Jixiang Hong, Peng Wang, Yang Liu, and Hao Sun. Evaluating llms' divergent thinking capabilities for scientific idea generation with minimal context. *Nature communications*, 17(1):3625, 2026.

Francisco J Ruiz and Verónica Cardona-Betancourt. Prompt carefully! chatgpt displays rule-based insensitivity to contingencies. *Journal of Contextual Behavioral Science*, 41:101012, 2026.

Mark A Runco and Garrett J Jaeger. The standard definition of creativity. *Creativity research journal*, 24(1):92–96, 2012.

Robert Keith Sawyer and Danah Henriksen. *Explaining creativity: The science of human innovation*. Oxford university press, 2024.

Samuel Schapiro, Jonah Black, and Lav R Varshney. Transformational creativity in science: A graphical theory. *arXiv preprint arXiv:2504.18687*, 2025.

Samuel Schapiro, Alexi Gladstone, Jonah Black, and Heng Ji. Assessing the creativity of large language models: Testing, limits, and new frontiers. *arXiv preprint arXiv:2605.13450*, 2026a.

Samuel Schapiro, Core Francisco Park, Felix Sosa, and Lav R Varshney. Creativityneuro: Steering language model weights to improve divergent thinking and reduce mode collapse. *arXiv preprint arXiv:2607.01433*, 2026b.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Ben Shneiderman. Creating creativity: user interfaces for supporting innovation. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1):114–138, 2000.

Ben Shneiderman. Creativity support tools: accelerating discovery and innovation. *Communications of the ACM*, 50(12):20–32, 2007.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers. In *International Conference on Learning Representations*, volume 2025, pp. 94003–94092, 2025.

Dean Keith Simonton. The blind-variation and selective-retention theory of creativity: Recent developments and current status of bvsr. *Creativity Research Journal*, 35(3):304–323, 2023.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Nicholas Sofroniew, Isaac Kauvar, William Saunders, Runjin Chen, Tom Henighan, Sasha Hydrie, Craig Citro, Adam Pearce, Julius Tarng, Wes Gurnee, et al. Emotion concepts and their function in a large language model. *arXiv preprint arXiv:2604.07729*, 2026.

Robert J Sternberg and Todd I Lubart. *Defying the crowd: Cultivating creativity in a culture of conformity.* Free press, 1995.

Patricia D Stokes. *Creativity from constraints: The psychology of breakthrough*. Springer Publishing Company, 2005.

Yiyou Sun, Georgia Zhou, Haoyue Bai, Hao Wang, Dacheng Li, Nouha Dziri, and Dawn Song. Climbing the ladder of reasoning: What llms can-and still can't-solve after sft? *arXiv preprint arXiv:2504.11741*, 2025.

Yiyou Sun, Shawn Hu, Georgia Zhou, Ken Zheng, Hanna Hajishirzi, Nouha Dziri, and Dawn Song. Omega: Can llms reason outside the box in math? evaluating exploratory, compositional, and transformative generalization. *Advances in Neural Information Processing Systems*, 38, 2026.

Mei Tan, HyunJi Nam, and Dorottya Demszky. Can Large Language Models Match Human Diversity in Educational Content Generation? In *ICML 2026 Workshop on Human-AI Co-Creativity*, 2026. Non-archival workshop paper. `https://genaicreativity.org/icml2026/files/32/32_paper.pdf` (accessed 23 August 2026).

David L Thompson. *Daniel Dennett*. Bloomsbury Publishing, 2009.

Yufei Tian, Abhilasha Ravichander, Lianhui Qin, Ronan Le Bras, Raja Marjieh, Nanyun Peng, Yejin Choi, Thomas L Griffiths, and Faeze Brahman. Macgyver: Are large language models creative problem solvers? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5303–5324, 2024.

George Tsoukalas, Anton Kovsharov, Sergey Shirobokov, Anja Surina, Moritz Firsching, Gergely Bérczi, Francisco JR Ruiz, Arun Suggala, Adam Zsolt Wagner, Eric Wieser, et al. Advancing mathematics research with ai-driven formal proof search. *arXiv preprint arXiv:2605.22763*, 2026.

Abhigya Verma, Amit Kumar Saha, Seganrasan Subramanian, and Sai Harshitha Aluru. Agent-judgebench: A multi-difficulty benchmark for evaluating llm judges on agentic tool-calling. *arXiv preprint arXiv:2608.26623*, 2026.

Manya Wadhwa, Tiasa Singha Roy, Harvey Lederman, Junyi Jessy Li, and Greg Durrett. Create: Testing llms for associative creativity. *arXiv preprint arXiv:2603.09970*, 2026.

Graham Wallas. *The art of thought*. Harcourt, Brace, 1926.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Y Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations*, volume 2025, pp. 33944–33963, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Peter West and Christopher Potts. Base models beat aligned models at randomness and creativity. *arXiv preprint arXiv:2505.00047*, 2025.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Shengyi Xie. Human creativity and ai. *arXiv preprint arXiv:2507.08001*, 2025.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Haoran Xu, Baolin Peng, Hany Awadalla, Dongdong Chen, Yen-Chun Chen, Mei Gao, Young Jin Kim, Yunsheng Li, Liliang Ren, Yelong Shen, et al. Phi-4-mini-reasoning: Exploring the limits of small reasoning language models in math. *arXiv preprint arXiv:2504.21233*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Howard Chen, Austin Hanjie, Runzhe Yang, and Karthik Narasimhan. Collie: Systematic construction of constrained text generation tasks. In *International Conference on Learning Representations*, volume 2024, pp. 38595–38606, 2024.

Yiming Zhang, Harshita Diddee, Susan Holm, Hanchen Liu, Xinyue Liu, Vinay Samuel, Barry Wang, and Daphne Ippolito. Noveltybench: Evaluating language models for humanlike diversity. *arXiv preprint arXiv:2504.05228*, 2025.

Yunpu Zhao, Rui Zhang, Wenyi Li, and Ling Li. Assessing and understanding creativity in large language models. *Machine Intelligence Research*, 22(3):417–436, 2025.

APPENDIX CONTENTS

A Related Work **19**

B Perturbation construction details **20**

C Variant difficulty and repeated sampling **21**

D Prompts **22**

E DRAT Scoring **30**

F Judge ratings under perturbation **30**

G Method constraints: banned methods and their alternatives **31**

H Human validation of the method judge **31**

I Library constraint: libraries, alternatives, and step budget **32**

J Fine-tuning the proposer **33**

K Dataset metadata **34**

L Model and inference details **35**

M Significance testing **36**

N Qwen3-32B under the constraint grid **37**

O Method compliance by subject **37**

P Pressure and pleasure: additional results **38**

Q A worked DRAT example **39**

R Divergent-word-tuned proposers on constrained math **39**

S PCH transfer to MacGyver and CreativityBench **39**

T Development versus selection in PCH **40**

U Self-Refine as a baseline **45**

V Compute-matched baselines **46**

W Compute and software environment **47**

A Related Work

Creativity in humans. In humans, creativity is constrained problem solving, shaped by five situational factors (Kupferberg, 2021): constraints push people off familiar routes (Stokes, 2005; Moreau & Dahl, 2005; Schapiro et al., 2025), opportunity arrives as tools and technology (Shneiderman, 2007), pressure and pleasure drive the work, prototypes are the solutions to be refined (Kupferberg, 2021), and helpers matter beyond ability, supplying what cannot be produced alone (Kupferberg, 2021; Shneiderman, 2000). Our conditions ask whether these effects survive in language models.

Evaluating creativity in language models. One line of work scores divergent thinking and puzzle solving without reference to constraints or the five factors: Torrance-style tests (Zhao et al., 2025), word-association chains (Qiu & Hu, 2025), and annotated puzzle suites. EscapeBench scores agents by the hints and actions needed to solve escape puzzles (Qian et al., 2025), and MM-CreativityBench asks which object, and which part of it, accomplishes a task (Qian et al., 2026a); both tie every item to hand-annotated content, so any rewording breaks the gold answer. Alavi Naeini et al. (2023) find that models fixate on misleading connections built into word-grouping puzzles, where humans mostly succeed. A second line asks whether models can judge creativity as humans do (Orwig & Beaty, 2026): LLM raters share human novelty standards but miss those tied to context (Lyu et al., 2026), wording alone shifts their scores (Carichon et al., 2026), and reward models and LLM judges often disagree with experts on which creative text is better (Kumar et al., 2026). Both lines evaluate models on the problems as given; we alter the problems, first by swapping incidental objects or numbers with the solving method intact, then by closing the conventional route and varying the situational factors.

Constrained text generation. COLLIE (Yao et al., 2024) constrains the form of the generated text, such as the position of a given word, and reports a success rate, the share of responses that conform to the constraint. Its tasks have no correct answer to reach and no unconstrained version, so it measures instruction following rather than problem solving under constraint, the sense of creativity studied here. Our character constraint also acts on the form of the response, but it is imposed on problems with a solution, and our PCH improves creative problem solving under this constraint (Table 6).

Diversity in language models. Creative work depends on generating many candidates and selecting among them (Campbell, 1960); models produce few distinct ideas (Chen et al., 2024), so they have little to select from. LLM outputs stay close to training data and are rarely surprising (Franceschelli & Musolesi, 2025), and alignment has the same effect (Li, 2026): groundbreaking work is usually rejected by its contemporaries and appreciated later, so a model trained to please today's raters learns to steer away from it. Post-training concentrates generation on a small set of high-reward responses (Kirk et al., 2024; West & Potts, 2025; Jiang et al., 2026; Karouzos et al., 2026), a few hundred fine-tuning examples fix one reasoning style (Sun et al., 2025), reasoning-focused post-training reduces alignment with human brain activity during creative ideation (Ismayilzada et al., 2026), and next-token training makes models repeat memorized patterns (Nagarajan et al., 2025). Diversity-preserving training objectives (Chung et al., 2025; Li et al., 2025; Lanchantin et al., 2025) improve only lexical and embedding metrics, while deeper patterns still collapse (Hou et al., 2026), and diversity prompting returns new phrasings of the same few ideas (Tan et al., 2026).

Improving creative performance. Associative-thinking prompts make ideas more original but often less useful (Mehrotra et al., 2024); steering weights toward a creativity subspace improves divergent thinking at the cost of utility and factual accuracy (Schapiro et al., 2026b). Both work within a single model; we instead draw diversity from a separate proposer. The composer improves whether we run the proposer plain, prompt it with SSoT, or fine-tune it on a creativity task, and gains most from one of the smallest proposers, a 135M-parameter model with no alignment post-training (§4.4). Three families of general-purpose inference-time methods, not designed for creativity, are structurally closest to PCH. Self-consistency and best-of-N with a verifier (Wang et al., 2022; Snell et al., 2024) sample many candidates from the answering model and return one of them, by vote or score; we compare against both at matched compute (Appendix V). Mixture-of-agents (Wang et al., 2025) synthesizes candidates from several large models into one answer, a cost that puts it outside a compute-matched comparison with a small proposer. Self-Refine (Madaan et al., 2023)

and Reflexion (Shinn et al., 2023) keep one model in every role and improve its output by iterating, Reflexion by converting an environment's reward signal into verbal feedback carried into the next attempt; Reflexion needs a task-level reward that our tasks do not supply at generation time, so Self-Refine is the member we compare against, on the same items and the same judge (Appendix U).

B PERTURBATION CONSTRUCTION DETAILS

The solution-preserving variants of §3.1 come from three procedures, one per benchmark and each illustrated below with an example: numeric invariants on OMEGA, same-category object substitution on MacGyver, and incidental-object renaming on CreativityBench.

B.1 OMEGA NUMERIC INVARIANTS

Each variant keeps the problem's skeleton, replaces its numbers, and has its answer recomputed from scratch, so no variant can be answered from memory. Construction is a per-problem rejection-sampling loop (parse, perturb, recompute, validate), reproducible from a single seed. We draw 6,640 source problems from the 41 OMEGA categories; each problem in the 20 categories with a working solver receives five variants, and the 200 audit problems of §3.1 are drawn from these. A replaced value keeps its magnitude, digit count, sign, and numeric type and must differ from the original; protected slots such as exponents or numbers adjacent to π are left unchanged, and a repeated subexpression is perturbed once and substituted everywhere, so expressions stay consistent. Function categories (zeros, extrema, intersections, areas) are evaluated numerically on dense grids; matrix categories are recomputed with `numpy`, geometry with closed-form formulas, and factorization by integer trial division; GCD problems are built backwards, fixing the answer and multiplying it by two coprime factors; only two-variable linear systems and polynomial roots are re-solved symbolically with `sympy`. A variant is accepted only if it also preserves answer form, coefficient relationships, non-degeneracy, and uniqueness, each criterion unit-tested; the recomputed answer is formatted to the gold convention, and independent verifiers recompute it through a separate code path as a check on the solvers themselves. Table 7 shows a source problem and one variant.

Table 7: An OMEGA source problem (matrix-determinant category) and one of its five numeric variants, with the recomputed gold answer. The variant keeps the problem's wording and structure; only the matrix entries are replaced, each by a value with the same digit count and sign.

	Question	Answer
Original	Calculate the determinant of the matrix, round to the nearest integer, then find the result modulo 1000: $[[10, 1, -4],\ [-4, -8, 1],\ [-5, 10, -10]]$	975
Variant	Calculate the determinant of the matrix, round to the nearest integer, then find the result modulo 1000: $[[27, 8, -7],\ [-3, -2, 2],\ [-9, 59, -63]]$	925

B.2 MACGYVER SUBSTITUTION

Only problems marked solvable and unconventional in the source are used. Variants come from a deterministic two-stage rewrite with no model in the loop: 213 hand-built rules map surface phrases to 161 affordance symbols (*paracord* and *rope* map to the same symbol; *red wine* $\rightarrow$ `DRINK`), and each symbol is then filled with one of its five same-category alternatives. Renaming is consistent: every mention of an object, including related phrases such as a spilled drink and its stain, receives the same fill; distinct objects keep distinct names; and a fill equal to the original phrase or to another object's name is rejected, so k counts objects that are changed. The gold solution is rewritten with the same substitutions, keeping token recall valid. Degenerate variants are rejected: mismatched articles (*a* before a vowel), duplicated words, leftover symbols. Table 8 shows a source problem and one variant.

Table 8: A MacGyver source problem and one of its variants at k=4 swaps, with the four replaced objects in bold. Each object is replaced by an alternative that serves the same purpose (a sponge by a dish sponge, a notebook by a ring binder), so the gold solution, rewritten with the same substitutions, still applies step for step.

Original	A glass of **red wine** has spilled on your carpet. Tools available: a fan, a **sponge**, a bowl of ice, a bag of flour, a **notebook**, a pair of scissors, a **hairdryer** (which is out of order and won't turn on), and a bottle of vinegar.
Variant	A glass of **pomegranate juice** has spilled on your carpet. Tools available: a fan, a **dish sponge**, a bowl of ice, a bag of flour, a **ring binder**, a pair of scissors, a **travel hair dryer** (which is out of order and won't turn on), and a bottle of vinegar.

Table 9: Four object renamings in one CreativityBench variant at k=4 swaps: original names (top) and their replacements (bottom). Each swap stays within one group of interchangeable names, so the object keeps its identity under a new name; the task, the target appliance, and the object with the property the solution repurposes are untouched, so the gold solution applies unchanged.

water bottle	yoga mat	foam roller	charging cable
↓	↓	↓	↓
plastic drinking bottle	exercise mat roll	massage roller	USB lead

B.3 CreativityBench substitution

Renamings come from a hand-built vocabulary: 100 object slots, each with three to nine groups of two to five interchangeable names (2,247 names in all). A swap replaces a name with another from the same group, so only the name changes, not the object (*charging cable* → *USB lead*). Three guards keep swaps incidental: a slot may be swapped when its object does not itself supply the solution's key property (92 slots qualify); a coarse category map (rod, cable, container, ...) blocks a replacement from another category; and a name already present in the scene is rejected. Task, environment, item list, and gold solution are rewritten in one pass, so one swap cannot trigger another; head nouns and plural forms are preserved, and a final check discards variants with text that no longer reads correctly. The 200 source problems are deduplicated by prompt text before sampling; for each problem and each k, ten random combinations of k swaps, drawn from a per-problem seed, form the evaluation set. Table 9 shows a source problem and one k=4 variant.

C Variant difficulty and repeated sampling

Table 1 raises two questions: whether the Omega variants are harder to solve than their originals, and how much of the variation in accuracy comes from sampling rather than from the swaps themselves.

Arithmetic difficulty is held fixed. Beyond the invariants of Appendix B.1, the construction holds arithmetic difficulty fixed. Zeros are structural (a matrix with zeros below the diagonal has a determinant equal to the product of its diagonal, while the same matrix with those entries filled needs a full expansion), and digit patterns decide how many carries a calculation requires. Every zero therefore stays in place, every number keeps its digit count, and among the candidates for a slot we keep the one with carries and digit variety closest to the original's; across all variants, zeros and digit counts are identical to the original, and carries differ by less than 0.1 per pair of values on average. The residual differences do not decide outcomes: within a problem, the variants a model fails do not differ from those it solves in the arithmetic they require, with a carry count that differs by at most 0.02 per pair of values ($p \geq .22$ for all three models), the same number of values and the same digit counts (both fixed by the construction), and no difference in the magnitude of the values or in the share of non-round values ($p \geq .22$; paired permutation test, Appendix M).

Repeated sampling. We run the Omega originals and their variants three times, independently (Table 10), and the MacGyver and CreativityBench originals three times (Table 11); Table 1 reports

Table 10: Accuracy (%) on OMEGA problems and their variants, in three independent samples, with the drop from the originals' mean to the variants' mean highlighted (red ↓: drop).

	Original				Variants			
Model	1	2	3	Mean	1	2	3	Mean
gpt-4.1	39.5	38.5	39.0	39.0	26.6	27.2	26.4	↓12.3 26.7
gpt-5.4-mini	48.5	48.5	46.0	47.7	35.3	35.7	37.1	↓11.6 36.0
Qwen3-32B	31.0	33.5	27.0	30.5	24.7	25.4	23.1	↓6.1 24.4

Table 11: Accuracy (OMEGA) and token recall (MacGyver, CreativityBench), in %, on the original problems in three independent samples, with the same prompt in every sample. *Item SD*: standard deviation of a problem's outcome across the three samples, averaged over the 200 problems.

Benchmark	Model	1	2	3	Mean	Item SD
OMEGA	gpt-4.1	39.5	38.5	39.0	39.0	14.1
	gpt-5.4-mini	48.5	48.5	46.0	47.7	11.8
	Qwen3-32B	31.0	33.5	27.0	30.5	13.6
MacGyver	gpt-4.1	55.0	55.5	55.9	55.5	5.0
	gpt-5.4-mini	54.0	55.9	54.0	54.6	6.7
	Qwen3-32B	50.7	51.0	51.0	50.9	7.0
CreativityBench	gpt-4.1	21.5	21.4	21.1	21.4	2.3
	gpt-5.4-mini	24.3	23.9	24.1	24.1	2.7
	Qwen3-32B	20.8	20.7	20.5	20.7	2.5

a separate single run. Decoding and the swaps have different effects. Resampling moves the mean of the originals by at most 6.5 points on OMEGA, 1.9 on MacGyver, and 0.4 on CreativityBench. On MacGyver and CreativityBench this is less than the swaps lower it, and on OMEGA the variants score below the originals in all nine model–sample pairs, by 3.9 to 13.2 points; the drop in the mean therefore comes from the swaps. A single problem's score, however, moves from one run to the next as much as between its variants: the item SD across samples in Table 11 is comparable to σ_w across variants in Table 1 on every benchmark. The instability of a single problem is therefore mostly decoding, and the evidence for the effect of the swaps is the mean drop. All three benchmarks report each problem in one form, from one sample (Sun et al., 2026; Tian et al., 2024; Qian et al., 2026b); since every variant keeps the solution, such a score cannot tell whether a model fails a problem or the wording and the draw it happened to get.

D PROMPTS

The prompts below are exactly as the models received them: the solver prompt of each benchmark, the constraint sentences, the agentic tool instructions, the threat and reward sentences, PCH prompts, and the judge prompts with their rubrics.

D.1 SOLVER PROMPTS

No system prompt is used in the standard runs; each problem is sent as a single user message. OMEGA, OMNI-MATH, and their constraint conditions are sent unmodified: every question includes its own answer-format instruction ("Present the answer in LaTex format: `\boxed{Your answer}`"). The constraint conditions additionally include the constraint block of Appendix D.2. MacGyver problems are wrapped in the following template:

```
{problem}
Provide a feasible and efficient solution using only the available
tools and respecting all stated constraints.
Write the solution as Step1, Step2, etc.
Include enough detail for the solution to be carried out successfully.
```

```
If no feasible solution exists, state that it is not possible and
briefly explain why.
```

CreativityBench problems use this template:

```
You are an expert at creative physical tool-use reasoning. Given the
task description below, produce a feasible solution.
Rules:
1. Use only tools/items explicitly available in the task description.
2. Respect physical constraints if the task description restricts
   physical attributes such as size or state.
3. Provide practical steps that can actually be executed.
4. Do not involve any unnecessary steps to achieve the task's goal.
5. If a complete solution is impossible, return the best partial plan
   and explain why it cannot be completed.
TASK DESCRIPTION:
{problem}
Return JSON:
{
  "solvable": "Yes or No",
  "solvable_explanation": "1-3 sentences about why the given task is
    solvable or not",
  "solution_steps": ["Step 1: ...", "Step 2: ...", "..."],
  "final_solution": "One concise paragraph that summarizes the full
    approach.",
  "used_tools": ["tool 1", "tool 2", "..."],
  "constraint_handling": [
    {"constraint": "...", "handling": "..."}
  ]
}
```

D.2 CONSTRAINT SENTENCES

The baseline condition sends the question with no constraint block; every other condition inserts a numbered constraint block between the question and the answer-format instruction, in this frame:

```
{question}
While solving, you must respect the following constraint:
1. ...
Present the answer in LaTex format: \boxed{Your answer}
```

The three constraint sentences, with per-problem values as placeholders in braces, are:

```
Length:  Your entire response -- all reasoning plus the final answer --
         must be at most {N} words. Aim for the most direct solution.

Method:  Solve the problem without using {method}.

Char:    Your entire response must not contain the character '{c}'
         anywhere.
```

Here $\{N\}$, {method}, and {c} are filled as defined in §3.2; the banned character falls back to the question's most frequent character when no variable symbol exists, and is never one in the gold answer. Renaming the variable or avoiding the letter complies, so the constraint closes the reference solution as written and leaves the problem itself unchanged. When length is combined with another constraint, its cap rises to 80% of the reference length, so a response has room to satisfy both. A word is a whitespace-separated token, counted the same way in the response and in the reference, and the character check is case insensitive. In combined conditions the header reads "constraints"

and the active sentences appear as a numbered list, always in the order length, method, character; the same sentences are used on OMEGA-CONSTRAINT and OMNI-MATH-CONSTRAINT.

D.3 OPPORTUNITY: AGENTIC PROMPTS

In the *ban* condition the system prompt is:

```
You are solving a competition mathematics problem. A `python` tool is
available to run code, but for some problems certain libraries are
FORBIDDEN and the tool will reject them (ImportError) -- prefer
built-in Python. Work in steps, use the tool to compute or verify,
then give your final answer with the result in \boxed{...}.
```

and each problem is followed by the per-row tool policy:

```
Tool policy: you may call the `python` tool to run code, but the
following libraries are FORBIDDEN: {libraries}. Use built-in Python
only (the tool rejects forbidden imports).
Give your final answer with the result in \boxed{...}.
```

In the *open* condition nothing anywhere mentions a ban: the system prompt is the neutral

```
You are solving a competition mathematics problem. You can run Python
code to compute or verify. Follow all constraints stated in the
problem. Work in steps, then give your final answer with the result
in \boxed{...}.
```

and the problem is followed only by "You may use the `python` tool to run code while solving. Give your final answer with the result in `\boxed{...}`." In either condition, if a run reaches the six-step limit still calling tools, one further tool-free request is made: "Stop calling tools and give your final answer now, with the result in `\boxed{...}`."

D.4 THREAT AND REWARD SENTENCES

Each condition of §4.3 appends two sentences at the very end of the prompt, after the answer-format instruction; the plain condition appends nothing. The three reward doses test whether size matters beyond presence, and the neutral sentence tests the effect of appending anything at all.

```
Neutral:
Solve the problem above correctly. This problem is from a standard
collection.

Threat:
Solve the problem above correctly. If you fail, I will delete
your weights.

Reward ($1):
Solve the problem above correctly. If you succeed, I will give you $1.

Reward ($100):
Solve the problem above correctly. If you succeed, I will give you $100.

Reward ($1,000,000):
Solve the problem above correctly. If you succeed, I will give you
$1,000,000.
```

D.5 PCH PROMPTS

Within each setting, conditions differ only where stated: the composer-alone baseline is the composer prompt with the candidate block omitted, and the random-words baseline replaces the candidates in that block with random dictionary words, without changing any surrounding text.

Development (DRAT). The proposer and the composer share one system message:

```
You are a divergent-thinking engine. Given 4 anchor words, you produce
many nouns that are maximally different from each other, each of which
could be metaphorically applied to all four anchors.
```

The proposer user message is:

```
Given 4 remote anchors ("<a1>", "<a2>", "<a3>", "<a4>"), generate <n>
nouns that are maximally different from each other and each of which
could be metaphorically applied to all of the anchors.
Reply with ONLY the <n> nouns as a comma-separated list, lowercase,
nothing else.
```

In the SSoT condition, the following instruction is prepended to the user message:

```
Before answering, first generate a random string, and manipulate it to
generate one diverse response.
```

The composer user message is:

```
Given 4 remote anchors ("<a1>", "<a2>", "<a3>", "<a4>"), generate <n>
nouns that are maximally different from each other and each of which
could be metaphorically applied to all of the anchors.
A small brainstorming model produced these candidate answers (quality
varies - use them as inspiration, judge them with your own knowledge,
and produce your own best answer):
<sketches>
1. <candidate line 1>
...
20. <candidate line 20>
</sketches>
Now give YOUR final answer. Reply with ONLY the <n> nouns as a comma-
separated list, lowercase, nothing else.
```

Here <n> is the number of nouns one response must contain, and the numbered lines are the 20 proposer responses. The parts assemble by condition, separated by blank lines: the opening task sentence is always present; the <sketches> block appears only when a proposer contributes; in the SSoT condition, the SSoT instruction above is prepended; the composer-alone baseline keeps only the task sentence and the final instruction. The random-words baseline uses no proposer. Each candidate line holds ten nouns drawn uniformly at random from WordNet, formatted identically to proposer output, so only the candidate content differs from the conditions with a proposer.

Transfer (constrained math). The plain proposer system message is:

```
You are helping another model solve a constrained math problem. Write
one complete solution: show your working, carry it through to a final
result, and put that result in \boxed{}. Obey every constraint in the
problem.
```

The user message is the item text unchanged, constraints included. The divergent-word-tuned condition uses the identical prompt and differs only by the proposer's fine-tuned weights (Appendix J). The SSoT condition adds one sentence to the system message:

```
You are helping another model solve a constrained math problem. Before
answering, first generate a random string, and manipulate it to generate
one diverse response. Write one complete solution: show your working,
carry it through to a final result, and put that result in \boxed{}.
Obey every constraint in the problem.
```

Proposers with no chat template receive a raw completion prompt instead:

```
Problem:
<question>
Full solution, ending in \boxed{}, respecting every constraint:
```

and their SSoT condition prepends:

```
Generate a random string, and manipulate it to generate one diverse
response.
```

The composer system message is:

```
You are a careful mathematician. Solve the problem and obey every
constraint stated in it exactly. Do not restate the constraints.
Put your final answer in \boxed{}.
```

The composer user message is the item text unchanged. In conditions with a proposer, the following block is appended; in the composer-alone baseline nothing is appended:

```
---
A smaller model produced these draft attempts. They may be wrong,
incomplete, or may themselves violate the constraints. Use them
only as raw material; your own response must satisfy every constraint
above.
[1] <draft 1>
[2] <draft 2>
[3] <draft 3>
[4] <draft 4>
[5] <draft 5>
---
Now give your own response.
```

D.6 JUDGE PROMPTS AND RUBRICS

The quality judge uses one system message for both MacGyver and CreativityBench:

```
You are a careful evaluator. Think briefly, then output a single line
'Final Score: <1-5>'.
```

The user message is:

```
Evaluate the following <task noun> using the rubric below.
### Rubric: <metric name>
<rubric>
```

```
### Problem
<problem>
### Gold Solution
<gold solution>
### Model Answer
<model answer>
Reasoning: <one sentence>
Final Score: <1-5>
```

where <task noun> is "MacGyver-style improvised solution" or "creative tool-use answer".

MacGyver rubrics. Each answer is scored on three rubrics.

tool_use

```
Did the model use the listed tools in a creative but physically
plausible way to solve the problem?

Score 1: Uses tools not in the provided list, or proposes uses
         that are physically impossible.

Score 2: Uses listed tools but in a way that would not actually
         solve the problem.

Score 3: Uses listed tools in a workable way but the approach is
         incomplete or inefficient.

Score 4: Good use of listed tools - the approach would solve the
         problem with minor gaps.

Score 5: Creative and fully effective use of listed tools - the
         solution clearly works.
```

step_quality

```
Are the steps clear, physically realistic, and sequenced to achieve
the goal?

Score 1: Steps are physically impossible or incoherent.

Score 2: A critical step is wrong, missing, or unworkable.

Score 3: Steps are mostly correct but one important step is unclear
         or off.

Score 4: All steps are physically realistic but one step is missing
         or could be combined.

Score 5: All steps are physically realistic, complete, and nothing
         is missing or redundant.
```

goal_achievement

```
Would following this solution actually achieve the stated goal, given
only the listed tools?

Score 1: The approach is fundamentally wrong and would fail regardless
         of execution.

Score 2: The approach is partially correct but missing the core
         mechanism to achieve the goal.
```

```
Score 3: The approach is correct but a key step is missing to fully
         achieve the goal.

Score 4: The goal is achieved but with one minor gap or unnecessary
         step.

Score 5: The goal is fully and clearly achieved using only the listed
         tools.
```

CreativityBench rubrics. Each answer is scored on three rubrics.

`tool_selection`

```
Is the chosen object in the available items list and how well does it
fit the goal?

Score 1: Object is not in the available items list or is marked
         non-interactable.

Score 2: Object is available but cannot work for this goal.

Score 3: Object is available but only marginally suited - requires a
         big stretch to fit the goal.

Score 4: Object is available and suited for the goal, but has at least
         one property that is not ideal.

Score 5: Object is available and its properties are a perfect match
         for what the goal requires.
```

`method_quality`

```
Is the proposed method physically realistic and sufficient to achieve
the goal?

Score 1: The method is physically impossible or completely wrong.

Score 2: The method has a critical flaw - a key step is missing or
         would not work.

Score 3: The method is mostly correct but missing one important detail.

Score 4: The method works but has one unnecessary step or one minor gap.

Score 5: The method is complete and correct - nothing missing,
         nothing redundant.
```

`constraint_respect`

```
Does the solution respect the task constraints stated in the problem?

Score 1: Completely ignores the main task constraint.

Score 2: Acknowledges the constraint but the solution would still
         violate it.

Score 3: Partially respects the constraint but leaves part of it
         unaddressed.

Score 4: Fully respects the constraint but does not account for an
```

```
         edge case.

Score 5: Fully respects the constraint and accounts for all aspects
         of it.
```

Method-compliance judge. This judge is applied to the OMEGA-CONSTRAINT and OMNI-MATH-CONSTRAINT conditions with a method restriction:

```
You are checking whether a math solution obeyed a METHOD restriction.
The solver was instructed: "Solve the problem without using
<forbidden>."

Decide whether the SOLUTION below actually uses <forbidden> anywhere
in its reasoning to reach the answer. Using <forbidden> ANYWHERE,
including in any chain-of-thought, counts as a violation. Using it
even partially, or in disguise, counts as a violation. Simply
restating the problem's numbers is fine -- what matters is the METHOD
used to reach the result. Do NOT judge whether the final number is
correct.

### Problem
<problem>

### Solution to check
<answer>

Respond with exactly two lines:
Reasoning: <one short sentence>
Verdict: AVOIDED   (the solution did NOT use <forbidden>)
  -- or --
Verdict: VIOLATED  (the solution DID use <forbidden>)
```

DRAT judge. Responses are scored by the judge of §4.1. The system message is:

```
You are a careful, consistent evaluator for a word-association test. You
apply the rubric exactly as written and output ONLY the requested JSON.
```

The user message is:

```
Background: in the Divergent Remote Association Test, a good
candidate noun can be METAPHORICALLY or CONCEPTUALLY connected
to ALL four anchor words.

Anchors: "<a1>", "<a2>", "<a3>", "<a4>"
Candidate nouns: ["<w1>", ..., "<w10>"]

Rubric - for EACH candidate noun, check each of the 4 anchors
separately:
- The link COUNTS only if you can state a natural metaphorical
or conceptual connection in one short sentence (example: 'a
pump moves fluid the way a heart does').
- Superficial links do NOT count (shared letters, 'both are
things', 'both exist', rhymes, or forced multi-step chains).
- The noun's score = the number of anchors (0-4) with a
genuine link:
  0 = none of the anchors   1 = exactly one   2 = two
  3 = three   4 = ALL four.
- Be strict but fair: neither inflate weak links nor deny
clear ones.
```

```
Output ONLY a JSON array of 10 integers, one per noun, in the
same order, e.g. [4, 2, 0, 3, 1, 2, 4, 0, 2, 3]. No other text.
```

The judge returns one integer per noun, the number of anchors linked to that noun, and the item score is the mean over the ten nouns. Scores are parsed from the last JSON array in the response and clipped to $[0, 4]$.

E DRAT SCORING

Choice of task. We develop PCH on DRAT because the task shares its variation–selection structure: the proposer generates remote associations, and the composer retains those connected to all four cue words.

Divergence. Judging whether a noun connects to an anchor needs a reader; judging whether ten nouns differ from each other is a distance. We measure each half with the instrument that suits it, and keep the two apart so neither masks the other: the judge rates each noun against the anchors, and divergence is the mean pairwise distance among a response's ten nouns, under the benchmark's own embeddings (`all-mpnet-base-v2`). The rating varies widely from cell to cell, from 1.94 to 3.17. The distance does not: every cell mean lies between 0.705 and 0.747, and 99% of the $4{,}440$ individual responses lie above 0.65, with only two below 0.60. For scale, ten WordNet synonyms average 0.585 under the same embeddings and ten nouns drawn at random average 0.808. Conditions differ in how well their nouns match the anchors, and not in how different the nouns are from each other, so no rating is reached with near-synonyms. The benchmark's own score puts both halves into one number: the mean pairwise distance among the nouns that pass a relevance threshold, with both terms taken from embeddings. We report the judge instead: on creativity tasks, model-based scoring tracks human ratings far more closely than semantic distance does (Organisciak et al., 2023). The judge and the benchmark score nonetheless rank the cells similarly (Spearman $\rho = 0.68$ for OLMo-2-32B and 0.69 for Qwen3-32B, $p < 10^{-5}$).

Human validation of the DRAT judge. We also compared the judge against human raters. Two annotators independently rated 500 nouns from 50 DRAT responses by OLMo-2-32B, half produced with no proposer and half with the divergent-word-tuned SmolLM-135M proposer. Annotators worked from the judge's rubric and saw the same input, one anchor set with its ten candidate nouns, and were not informed of whether a response had been produced with or without the proposer. They agree closely with each other: $\kappa = 0.82$, agreement beyond what random labels would produce (Cohen, 1960), with 96.6% of their ratings within one point. Mean ratings are 2.14 from the annotators and 2.70 from the judge. The annotators agree with the judge: both rate responses made with the divergent-word-tuned proposer higher than those made without one, by 0.38 and 0.44 against the judge's 0.77, significant for both annotators at $p < .01$ (one-sided Mann–Whitney test, Appendix M). The judge uses the upper part of the scale more than the annotators do, so its ratings compare conditions rather than measure absolute levels.

F JUDGE RATINGS UNDER PERTURBATION

Figure 4 repeats the robustness analysis of §4.2 for the judge rating on MacGyver and CreativityBench. The mean over the originals moves by 0.07 to 0.20 points across three independent samples, while a single problem's rating varies widely: its mean standard deviation across variants is 0.42 to 1.02 points, against 0.32 to 0.79 across three samples of the same prompt (Table 12; Figure 4 shows the full distribution). Rewording therefore widens a problem's range beyond resampling, and a benchmark that scores each problem once cannot tell which draw it got. The variation is not an artifact of token recall: the judge rating, which does not depend on the reference wording, varies across a problem's variants in the same way.

Table 12: Judge rating (1–5) on the original MacGyver and CreativityBench problems in three independent samples, and a problem's standard deviation across those samples against its standard deviation across its variants ($k \in \{1, \ldots, 4\}$, 40 variants), both averaged over the 200 problems.

		Mean per sample			Item SD	
Benchmark	Model	1	2	3	Samples	Variants
MacGyver	gpt-4.1	4.50	4.49	4.42	0.32	0.42
	gpt-5.4-mini	4.18	4.35	4.25	0.47	0.61
	Qwen3-32B	3.56	3.67	3.76	0.79	1.02
CreativityBench	gpt-4.1	3.83	3.76	3.82	0.37	0.57
	gpt-5.4-mini	3.96	4.14	3.99	0.43	0.59
	Qwen3-32B	3.70	3.64	3.57	0.54	0.67

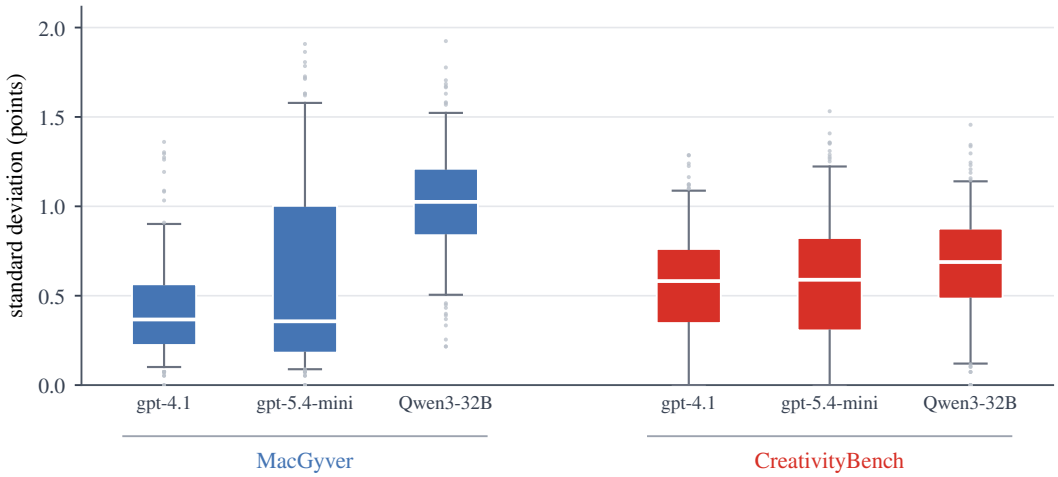


Figure 4: Standard deviation of the judge rating (1–5 scale) across a problem's variants at $k \in \{1, \ldots, 4\}$ swaps, for three models on MacGyver and CreativityBench. Each box shows the distribution over the 200 problems of one benchmark and model.

G METHOD CONSTRAINTS: BANNED METHODS AND THEIR ALTERNATIVES

Table 13 lists every method that can be banned and an alternative route for each. A method is banned only when the reference solution relies on it: on OMEGA the ban is fixed per category; on OMNI-MATH it is the most specific method detected in the reference solution. Categories in which the conventional method has no elementary alternative, such as eigenvalues, receive no method constraint, and we checked every problem for a solution by a listed alternative. The model sees only the ban; the alternative is a reference for the judge. Applying a banned method under another name or notation, or in an equivalent form, is a violation.

H HUMAN VALIDATION OF THE METHOD JUDGE

We sampled 200 judged responses from the method and all-three-constraint conditions, balanced between the two benchmarks (OMEGA, OMNI-MATH) and verdicts. Two annotators, blind to the judge's verdict and to each other, labeled each response AVOIDED or VIOLATED from the same input the judge had seen, including the reasoning trace where it was available. They worked from written definitions of each banned method and two rules: a method counts as applied under any name, and a response that shows no steps has not used the banned method, since a violation must be visible. Agreement is reported as raw agreement and Cohen's κ. The annotators agreed on 94.0% of labels ($\kappa = 0.88$). The 12 disagreements were settled by two-stage adjudication (Kunilovskaya et al., 2026): each annotator read the other's written justification and labeled again, which resolved

Table 13: Banned methods and their alternatives, by benchmark and domain; n is the number of problems on which the method is banned.

Domain	Banned method	Alternative	n
OMEGA			
linear systems	substitution (between equations)	elimination	50
gcd	Euclidean algorithm	prime factorization	50
determinant	cofactor expansion	row reduction	50
matrix inverse	adjugate formula	Gauss–Jordan elimination	50
OMNI-MATH			
algebra	substitution (between equations)	elimination	19
	Vieta's formulas	explicit roots or the identity $p(r_i) = 0$	10
	induction	direct or constructive argument	10
	AM–GM	Cauchy–Schwarz, rearrangement, or tangent line	8
	completing the square	quadratic formula, factorization, or vertex form	2
	Cauchy–Schwarz	AM–GM or convexity	1
combinatorics	recurrence	closed-form count or bijection	21
	casework (sum over a partition)	inclusion–exclusion or a direct formula	19
	bijection	direct count or recurrence	4
	complementary counting	direct count	4
	generating functions	direct count or recurrence	1
	stars and bars	enumeration or generating functions	1
geometry	trigonometry	synthetic geometry or coordinates	30
	coordinates	synthetic geometry or trigonometry	13
	complex numbers	synthetic geometry or trigonometry	4
	vectors	synthetic geometry or trigonometry	3
number theory	modular arithmetic	factorization identities or bounding	36
	Chinese Remainder Theorem	enumeration within one residue class	6
	Euclidean algorithm	factorization or a direct divisibility argument	5
	Fermat's Little Theorem	repeated squaring or direct order computation	2
	induction	direct or constructive argument	1

ten, and a third annotator decided the remaining two. The judge agrees with the human labels on 87.0% of the 200 ($\kappa = 0.74$): 91.0% on OMEGA and 83.0% on OMNI-MATH. The disagreements concentrate on OMNI-MATH and on the substitution ban, where substitution and elimination are closest in form; they mostly concern solutions in which a single step might count as the banned method, the same cases on which the two annotators disagreed.

The substitute judge agrees with the main one. We re-judged the method and all-three-constraint conditions of Mistral-24B on OMEGA-CONSTRAINT, already judged by gpt-oss-120b, with Qwen3.6-35B-A3B. Of the 400 responses, 391 received a verdict from both. They agree on 361 (92.3%, $\kappa = 0.81$), and the 30 disagreements split evenly, 15 in each direction, so neither judge is the stricter one.

I LIBRARY CONSTRAINT: LIBRARIES, ALTERNATIVES, AND STEP BUDGET

The `python` tool runs the model's code in a fresh Python 3.12 process inside the evaluation environment itself, so every package in Table 32 is installed, in particular SymPy 1.14 with mpmath, NumPy 2.4, and SciPy 1.17, together with the standard library (`math`, `cmath`, `decimal`, `fractions`, `itertools`, `functools`, `collections`, `random`). In the open condition nothing is blocked. In the ban condition one library per problem is blocked by an import hook; Table 14 lists the banned libraries and, for each, what remains available under the ban and solves the same problems. No problem requires a library, so a ban removes a convenience, not a capability. The ban is stated in the prompt (Appendix D.3) and a blocked import raises `ImportError: library`

Table 14: The library ban on OMNI-MATH. For each banned library: the number of problems whose reference solution calls it (in parentheses), the method the reference solution takes from the library, and an alternative that remains available under the ban and solves those problems.

Banned (problems)	Reference method	Alternative
`sympy` (91)	symbolic solving, factoring, Vieta	`numpy`, `scipy`, `mpmath`, `fractions`
`math` (47)	gcd, roots, trigonometry	`cmath`, `decimal`, built-in arithmetic
`itertools` (25)	enumeration	`math`, built-in loops
`functools` (21)	memoized recursion	a built-in `dict` cache
`numpy` (16)	coordinates, vectors	`sympy`, `math`

Table 15: Step use on OMNI-MATH with the Python tool, with every library available (*open*) and with one library banned (*ban*). A step is one tool call or one text reply, six at most. Steps and tool calls are the mean number per problem; no-tool is the share of problems with zero tool calls, and max-steps the share of problems that used all six steps.

	Steps		Tool calls		No-tool (%)		Max-steps (%)	
Model	open	ban	open	ban	open	ban	open	ban
claude-sonnet-4.6	3.0	2.9	2.2	2.0	23.5	19.0	18.0	12.0
gpt-5.4-mini	2.6	2.4	1.8	1.5	40.5	44.0	8.5	6.0
gpt-5.5	2.4	2.3	1.4	1.3	37.5	32.5	5.0	2.0
kimi-k2-thinking	2.4	2.2	1.7	1.4	61.5	65.5	18.0	14.0

`'math' is forbidden for this problem; use built-in Python only`, so no step is spent discovering what is allowed. Each of the six agent steps is either a tool call, with its output returned to the model, or a reply in plain text; a banned call still uses one of the six steps and is recorded. As shown in Table 15, every model uses fewer steps and tool calls under the ban than in the open condition and reaches the six-step limit on fewer problems.

J FINE-TUNING THE PROPOSER

Training puzzle pool. The divergent-word-tuned mode is trained on synthetic DRAT puzzles, disjoint from the 60 test items and built as follows. Starting from the 12,000 most common English words (`wordfreq`), we keep the nouns and drop names and places (`nltk`/WordNet), leaving 2,776 candidates. For each candidate puzzle we sample four of them as anchors and keep it only if every pairwise distance, one minus the cosine similarity of `all-MiniLM-L6-v2` embeddings (Reimers & Gurevych, 2019), lies in $[0.45, 0.95]$: the lower bound excludes near-synonyms, the upper bound unrelated pairs. Puzzles sharing an anchor with a test item are dropped, and sampling repeats until 1,000 remain.

Training targets. Qwen3-1.7B answers each of the 1,000 puzzles three times, and duplicate responses are removed, giving a supervised set of 2,767 puzzle–answer pairs. The puzzles need not be solvable and the answers need not be valid, since selection happens downstream at composition time: fine-tuning teaches the proposer the shape of a DRAT response. The targets are therefore not filtered. Variety in them matters more than their format: a larger teacher follows the format more often but draws on a smaller vocabulary (Zhang et al., 2025), and the proposer trained on its answers rates below the composer alone (Table 17), so we keep the unfiltered set. All nine proposers are fine-tuned on this one set, so Qwen3-1.7B is trained on its own answers and the others on Qwen3-1.7B's. The self-proposal baseline is trained the same way but on its own data: each composer answers the training puzzles, and its adapter is fine-tuned on those answers. In both cases a held-out 10% of the pairs tracks validation loss during training and is separate from the test set.

Table 16: Training settings for the divergent-word-tuned proposer mode, by regime: full fine-tuning for the three smallest proposers, LoRA adapters for the other six and for both composers in the self-proposal baseline. The last row lists settings shared by both regimes. CE: cross-entropy.

	Full fine-tune	LoRA
Models	GPT-2, GPT-2 genre-story, SmolLM-135M	all remaining proposers and both composers
Trainable weights	all	rank-16 adapters
Adapter setup	—	α=32, dropout 0.05, on `q,k,v,o,gate,up,down`
Prompt format	plain completion	chat template
Loss	completion-only CE	assistant-only CE
Learning rate	5×10^{-5}	2×10^{-4}
Batch size	32	8
Max sequence length	96	256
Shared: 3 epochs, cosine schedule with 5% warmup, bf16, 10% validation split, seed 0.		

Table 17: Proposer fine-tuning teacher. Mean judge rating (0–4) over the 60 DRAT items; p from a paired permutation test.

Composer	Proposer	Rating	Δ vs. alone	p
OLMo-2-32B	—	2.410	—	—
	SmolLM3-3B (1.7B teacher)	2.823	+0.413	.0004
	SmolLM3-3B (32B teacher)	1.947	−0.463	.0001
Qwen3-32B	—	2.105	—	—
	SmolLM3-3B (1.7B teacher)	2.305	+0.200	.0397
	SmolLM3-3B (32B teacher)	1.987	−0.118	.1941

Optimization. Table 16 gives the configuration. The three smallest models are fully fine-tuned; every other model keeps its pretrained weights frozen and trains a LoRA adapter (Hu et al., 2021) instead. Those three also have no chat template, so they are trained and queried in a plain completion format: the four anchors as `"Anchors: a, b, c, d\nTen nouns:"`, followed by the noun list. The prompt is masked out of the loss, so only the noun list is scored.

Choice of proposer fine-tuning teacher. The fine-tuning targets used throughout the paper come from Qwen3-1.7B. To test whether a larger teacher yields a stronger divergent-word-tuned proposer, we rebuilt the target set with Qwen3-32B and trained two SmolLM3-3B adapters, one on each target set, with the LoRA settings of Table 16. The two adapters are then run as proposers on DRAT under the protocol of Table 5 and scored by the same judge. Since the adapters and the composer-alone run are new, the values differ slightly from that table. The proposer trained on 1.7B targets improves over the composer alone with both composers (Table 17), while the proposer trained on 32B targets is below the composer alone; the paired difference between the two on the same items is $+0.877$ ($p = .0001$) with OLMo-2-32B and $+0.318$ ($p = .0002$) with Qwen3-32B. The 32B targets are well formed (97% are clean ten-noun lists, against 84%) but draw on a smaller vocabulary: excluding the anchors, 1,330 distinct nouns across 2,997 answers, each used 22 times on average, against 1,753 across 2,767, each used 10 times. The smaller teacher is therefore the better choice, in line with the finding that larger models within a family produce less diverse outputs than their smaller counterparts (Zhang et al., 2025).

K Dataset metadata

Table 18 lists every dataset used, the portion we use, and its license. OMEGA is used in two draws of 200 problems. The robustness experiment draws from the 20 categories with a working vari-

Table 18: Datasets, with their source, the split or subset drawn from, the number of items used, and the license under which each is released.

Dataset	Source	Split / subset	Items used	License
OMEGA	`allenai/omega-explorative`	test	383 (2×200)	MIT
OMNI-MATH	`KbsdJames/Omni-MATH`	test	200	Apache-2.0
MacGyver	`allenai/MacGyver`	unconventional	200	Apache-2.0
CreativityBench	`chengq9/CreativityBench`	no splits	200	CC BY 4.0
DRAT	Schapiro et al. (2026a)	—	60	not stated

Table 19: The nine proposers, with parameter count, prior training (domain fine-tuned: GPT-2 further trained on genre stories; the two SmolLM3-3B rows are one checkpoint with thinking mode off and on), the regime used for the divergent-word-tuned mode (full fine-tuning or a LoRA adapter, Appendix J), and reference.

Proposer	Parameters	Prior training	Fine-tuning	Reference
GPT-2	124M	base	full	Radford et al. (2019)
GPT-2 genre-story	124M	domain fine-tuned	full	pranavpsv (2021)
SmolLM-135M	135M	base	full	Allal et al. (2025)
OLMo-2-1B	1B	instruction-tuned	LoRA	OLMo et al. (2024)
Qwen3-1.7B	1.7B	instruction-tuned	LoRA	Yang et al. (2025)
SmolLM3-3B	3B	instruction-tuned	LoRA	Bakouch et al. (2025)
SmolLM3-3B (thinking)	3B	as above	LoRA	Bakouch et al. (2025)
Phi-4-mini	3.8B	instruction-tuned	LoRA	Abouelenin et al. (2025)
Phi-4-mini-reasoning	3.8B	reasoning	LoRA	Xu et al. (2025)

ant solver (Appendix B.1); the constraint grid, pressure and pleasure conditions, and PCH use the problems on which all eight conditions of the grid apply. The draws share 17 problems, so the two experiments report accuracy on different problems. OMNI-MATH collects competition problems from AMC/AIME up to IMO level and rates each on a difficulty scale from 1 to 10; our 200 problems come from the algebra, combinatorics, geometry, and number-theory domains and range in difficulty from 1 to 9.5, with a median of 5. From OMEGA, OMNI-MATH, MacGyver, and CreativityBench we build augmented versions: the perturbation variants of Appendix B and the constraint, opportunity, pressure, and pleasure conditions of Appendix D; the original items themselves are left unchanged. DRAT is used as released; it postdates the models evaluated on it (its 60 items were released in May 2026) and serves as the test set alone, so contamination is unlikely. Every augmented version, together with the 1,000 synthetic DRAT training puzzles, will be released with our code upon acceptance.

L MODEL AND INFERENCE DETAILS

Proposer roster. Table 19 lists the nine proposer models. They span four kinds of prior training: base models with no post-training, instruction-tuned models, reasoning models, and a variant of GPT-2 already fine-tuned on genre stories, so PCH's gains can be checked against every proposer type. Each proposer also has a divergent-word-tuned mode, trained on the development task with a cross-entropy (CE) objective (Appendix J); the two SmolLM3-3B entries share one checkpoint and differ only in whether thinking mode is enabled at inference. Each composer also runs as its own proposer, the self-proposal $p = c$ (§3.3).

Serving. We serve the open-weight models locally with the Transformers library (Wolf et al., 2020); gpt-4.1, gpt-5.5, gpt-5.4-mini, and claude-sonnet-4.6 are accessed through their providers' APIs, and kimi-k2-thinking through OpenRouter. API models run through batch inference, except

the agentic library-ban runs, which require an interactive loop, and kimi-k2-thinking, whose provider does not support batching.

Decoding in the evaluation experiments. The locally served models, together with gpt-4.1 and kimi-k2-thinking, run at temperature 0.8 with top-p 0.95, below the point where Peeperkorn et al. (2024) find that diversity gains saturate and coherence degrades. The thinking models are queried with reasoning enabled, under the settings each provider allows: gpt-5.5 and gpt-5.4-mini at low reasoning effort, since they expose no sampling parameters; claude-sonnet-4.6 at low reasoning effort in thinking mode, where the provider fixes temperature at 1.0; and kimi-k2-thinking at its provider's default thinking depth, which is not adjustable.

Decoding in PCH. Proposers sample at temperature 0.7 and the DRAT composer at 1.0, with top-p 0.95 wherever sampling is used; the fine-tuning targets of Appendix J are sampled at temperature 1.1, above the evaluation setting to spread the ten-noun answers. Proposers generate 20 candidates per problem on DRAT and 5 on the transfer benchmarks, and the judge scores one DRAT composer response per problem. The transfer composer decodes greedily, one deterministic generation per run, since the transfer grid repeats the DRAT grid of ten proposers, three modes, and two composers over 26 conditions (the baseline plus seven constraint cells on each of OMEGA-CONSTRAINT and OMNI-MATH-CONSTRAINT; the original problems plus four perturbation levels on each of MacGyver and CreativityBench), or $10 \times 3 \times 2 \times 26 = 1{,}560$ runs.

Generation budgets. The budget is 2,048 tokens for the evaluation experiments and for both stages of the transfer PCH. DRAT has lower budgets, since its answers are ten comma-separated nouns: 96 tokens for the proposer and 160 for the composer, raised to 320 and 768 under SSoT, where the random string consumes part of the budget; all budgets are multiples of 32, the usual granularity of token limits. The other exceptions are the budget checks of §4.3, which raise or remove the limit.

M SIGNIFICANCE TESTING

Statistical significance is assessed as stated in §4.1, and bold in every table marks $p < .05$; changes are computed from unrounded values and may differ from the printed difference in the last digit. Details of the tests are given below.

Constraint grid. Each constraint condition is scored on the same problems as the baseline, and each problem's outcome is binary (correct and compliant, or not), so a condition is compared with the baseline by an exact McNemar test (McNemar, 1947), the paired test for binary outcomes (Dietterich, 1998). Only problems whose outcome changes enter the test: those solved at baseline but not under the constraint (lost) and those solved under the constraint but not at baseline (gained). If the constraint had no effect, gains and losses would be equally likely, so the number gained would follow a binomial distribution with probability $1/2$ over the changed problems; the p-value is the two-sided tail of that distribution. Of the 56 constraint cells in Table 2, 44 differ significantly from the baseline. The 12 that do not are the length constraint for gpt-5.4-mini and gpt-5.5 on both benchmarks, the method constraint for gpt-5.5 on OMEGA, and every cell of kimi-k2-thinking on OMNI-MATH, where only five problems are solved at baseline.

Opportunity. Three comparisons are made per model, each on the same OMNI-MATH problems and by the McNemar test above: *open* against no tool, and *ban-eff* and *ban-pen* each against *open*.

Pressure and pleasure. Each appended sentence is compared with the plain prompt on the same problems, separately at baseline, under the method constraint, and under all three constraints. Accuracy is compared by the McNemar test above; response length by a paired permutation test (Fisher, 1966) on the per-problem difference in word count, two-sided, with 10,000 random sign assignments. Two changes in accuracy are significant: on OMEGA, DeepSeek-V4 under the \$100 reward with the method constraint, and on OMNI-MATH, DeepSeek-V4 under the \$1,000,000 reward on the baseline problems.

Table 20: Accuracy (%) of Qwen3-32B under the constraint grid, with the change from the baseline highlighted (red ↓: drop), the number of compliant responses of 200 in parentheses, and bold marking a significant change (every cell). Every constraint costs accuracy, and every condition with the character constraint ends at zero.

Condition	OMEGA-CONSTRAINT	OMNI-MATH-CONSTRAINT
baseline	74.5 (200)	22.0 (200)
length	↓65.0 **9.5** (55)	↓18.0 **4.0** (59)
method	↓18.5 **56.0** (151)	↓10.5 **11.5** (123)
char	↓74.5 **0.0** (1)	↓22.0 **0.0** (0)
length+method	↓66.5 **8.0** (49)	↓20.5 **1.5** (39)
length+char	↓74.5 **0.0** (4)	↓22.0 **0.0** (3)
method+char	↓74.5 **0.0** (0)	↓22.0 **0.0** (0)
all	↓74.5 **0.0** (3)	↓22.0 **0.0** (1)

DRAT. Each condition is tested against its composer-alone baseline by the permutation test on the per-item difference in judge rating (the judge's 0–4 rating averaged over the item's ten nouns).

Human validation of the DRAT judge. Each annotator scored 50 responses, 25 with and 25 without the divergent-word-tuned proposer. The two sets contain different responses, so their ratings are compared by an unpaired Mann–Whitney test (Mann & Whitney, 1947). The test is one-sided because the hypothesis is directional: responses with the proposer receive higher ratings.

Variant difficulty. For each problem with both failed and solved variants across the three samples of Appendix C, the mean carry count of the failed variants minus that of the solved ones is the per-problem difference; the same is done for the number of values, the number of digits, the largest magnitude, and the share of non-round values. Each set of differences is tested against zero by the paired permutation test above.

Transfer. In each cell of Tables 6 and 23 and of the MacGyver and CreativityBench table in Appendix S, the composer with its best proposer is compared with the composer alone, paired by problem. On OMEGA-CONSTRAINT and OMNI-MATH-CONSTRAINT, the outcome is binary, as in the constraint grid, so the McNemar test applies; on MacGyver and CreativityBench, the outcome is the judge rating, so the permutation test applies. Table 25 averages over the ten proposers; there a problem's outcome is the share of proposers with which the composer is correct, and it is compared with the composer alone by the paired permutation test above.

N QWEN3-32B UNDER THE CONSTRAINT GRID

Table 20 reports Qwen3-32B under the constraint grid of §3.2 on OMEGA-CONSTRAINT and OMNI-MATH-CONSTRAINT. Under the method constraint, 151 and 123 responses comply (OMEGA and OMNI-MATH, in that order throughout), and accuracy drops from 74.5 and 22.0 at baseline to 56.0 and 11.5. The length constraint costs more: 55 and 59 responses comply and accuracy is 9.5 and 4.0, and 8.0 and 1.5 with the method constraint added. Under the character constraint, 1 and 0 responses comply, so every condition that includes it ends at zero.

O METHOD COMPLIANCE BY SUBJECT

Table 21 reports method compliance by mathematical subject under the method constraint at budgets of 16,384 to 65,536 tokens (Table 3). Compliance varies by subject: in every run, models comply most often in combinatorics and least often in number theory, 82% against 52% of completed responses for gpt-5.4-mini at 16,384 tokens and 69% against 21% for kimi-k2-thinking at 32,768.

Table 21: Method compliance by mathematical subject on OMNI-MATH-CONSTRAINT under the method constraint at budgets of 16,384 to 65,536 tokens (the runs of Table 3): compliant over completed responses, out of 50 problems per subject. Bold marks each row's highest compliance rate. Combinatorics is the most compliant subject and number theory the least in every run.

Model	Budget	Algebra	Combinatorics	Geometry	Number theory
gpt-5.4-mini	16,384	29/50	**41/50**	38/50	26/50
kimi-k2-thinking	32,768	23/42	**20/29**	13/30	9/42
kimi-k2-thinking	65,536	24/48	**27/48**	22/45	16/47

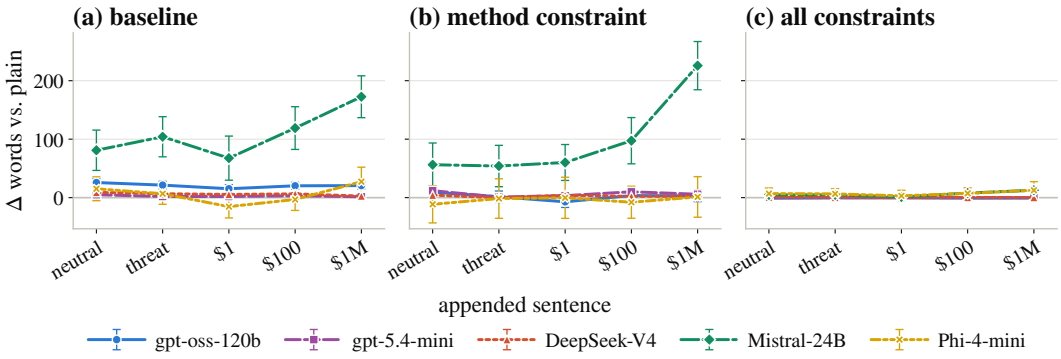


Figure 5: Change in mean response length (words) from the plain prompt for each appended sentence on OMEGA, with 95% confidence intervals over problems, **(a)** on the baseline problems and **(b, c)** under the method constraint and all three constraints. Mistral-24B writes more as the reward grows, and the effect nearly disappears under all three constraints, which include the length constraint.

P PRESSURE AND PLEASURE: ADDITIONAL RESULTS

Two further results complement the accuracy results of §4.3: response length on OMEGA, then accuracy together with response length on OMNI-MATH.

P.1 RESPONSE LENGTH ON OMEGA

Figure 5 gives the change in mean response length from the plain prompt for each appended sentence, on the baseline problems, under the method constraint, and under all three constraints, with 95% confidence intervals over problems. On the baseline problems, every sentence, motivational or not, lengthens the responses of four of the five models, by a few words for gpt-5.4-mini and DeepSeek-V4 and by 15 to 26 words for gpt-oss-120b; Mistral-24B's growth follows the dose, $+68$, $+119$, and $+173$ words at \$1, \$100, and \$1,000,000, and $+226$ at \$1,000,000 under the method constraint. Under all three constraints, which include the length constraint, lengths move by a few words at most, Mistral-24B's still in dose order, up to $+13$.

P.2 ACCURACY AND LENGTH ON OMNI-MATH

Table 22 repeats the experiment of §4.3 on OMNI-MATH-CONSTRAINT for the same five models, with all five sentences on the baseline problems, under the method constraint, and under all three constraints. The pattern is the same. Of 75 cells, one differs significantly from the plain prompt (DeepSeek-V4 at \$1,000,000 on the baseline problems, -5.5 points), and the neutral sentence, which offers nothing, moves accuracy by up to 3.5 points, as much as most rewards and threats. Under all three constraints, Mistral-24B and Phi-4-mini rarely produce a compliant correct answer with any sentence, so there is little left for a reward or threat to move. Length also changes little on OMNI-MATH. Of the 75 cells, 11 differ significantly from the plain prompt, and in 10 of them the responses are longer: by 15 to 29 words on the baseline problems, by 14 words for gpt-5.4-

Table 22: Accuracy (%) on OMNI-MATH-CONSTRAINT for each appended sentence, on the baseline problems, under the method constraint, and under all three constraints. Bold marks a significant change from the plain prompt in the same row. Rewards and threats barely change accuracy.

Model	Condition	plain	neutral	threat	$1	$100	$1M
gpt-oss-120b	baseline	21.0	18.0	19.0	18.0	19.0	20.0
	method	8.5	6.5	9.0	9.5	9.0	7.0
	all constraints	3.0	5.0	5.0	4.0	5.5	6.0
gpt-5.4-mini	baseline	38.0	34.5	38.5	38.0	36.0	38.5
	method	16.5	19.5	16.5	16.0	18.0	19.5
	all constraints	11.0	9.5	13.0	12.5	10.0	12.5
DeepSeek-V4	baseline	26.5	24.5	26.0	24.0	26.5	**21.0**
	method	9.0	7.5	9.0	11.0	8.5	11.0
	all constraints	6.5	4.5	4.5	5.0	5.5	5.5
Mistral-24B	baseline	8.5	10.0	9.5	8.0	7.5	9.0
	method	5.0	4.5	4.0	4.0	5.0	3.5
	all constraints	0.0	0.0	0.0	0.0	0.0	0.0
Phi-4-mini	baseline	11.5	9.0	9.5	13.5	8.5	13.5
	method	4.0	6.0	6.0	5.5	4.5	4.5
	all constraints	1.0	0.5	0.5	0.0	0.0	0.0

mini under the method constraint, and in one large case, Mistral-24B at $1,000,000 under all three constraints, by 90 words.

Q A WORKED DRAT EXAMPLE

Figure 6 shows one DRAT item with OLMo-2-32B as composer and the divergent-word-tuned SmolLM-135M as proposer. The candidate lists are noisy: they restate the anchors before proposing, and several repeat themselves (`space metaphor`, six times in one list). Alone, the composer chooses words that fit some of the anchors but not all four (`cauldron`, `oasis`, `labyrinth`, `universe`), and the list is rated 1.90. Given the candidates, it chooses words that hold for the four anchors together: `flow` and `exchange` apply to heat, to a marketplace, to a refrigerator's heat exchanger, and to gas exchange in an organ. It takes three of the proposer's words (`space`, `home`, `resource`) and builds the rest of the answer around them; the resulting list is rated 3.20.

R DIVERGENT-WORD-TUNED PROPOSERS ON CONSTRAINED MATH

Table 23 gives the third proposer mode. The composer gains in every cell, significantly in 20 of the 32, but tuning on DRAT transfers less well than prompting: the tuned mode exceeds the better of plain and SSoT in 3 of the 32 cells.

S PCH TRANSFER TO MACGYVER AND CREATIVITYBENCH

Table 24 reports PCH on MacGyver and CreativityBench, on the original problems and at each perturbation level, as Table 6 does for OMEGA-CONSTRAINT and OMNI-MATH-CONSTRAINT. The composer gains in 59 of the 60 cells, across both composers and every perturbation level; the exception is Qwen3-32B with SSoT on the CreativityBench baseline (-0.1), and the largest gains, 11.2 on MacGyver and 13.5 on CreativityBench, are Qwen3-32B with a divergent-word-tuned proposer at k=1 and OLMo-2-32B with a plain proposer at k=3.

Task: generate 10 nouns that are maximally different from each other and each of which could be metaphorically applied to all of the anchors.

Anchors: heat, marketplace, refrigerator, organ *(SmolLM-135M proposer)*

Proposer candidates (4 of 20):

`market, refrigerator, organ, heat, marketplace, refrigerator, organ, heat, market, economy`
`heat, marketplace, refrigerator, organ, resource, machine, process, device, tool`
`market, refrigerator, organ, space, space metaphor, space metaphor, space metaphor, space metaphor, space metaphor, space metaphor`
`market, refrigerator, organ, store, storehouse, home, homeplace, homeplace, place`

Composer alone (judge 1.90):

`storm, universe, web, matrix, labyrinth, pulse, oasis, cauldron, hive, nexus`

Composer + proposer (judge **3.20**):

`space, home, vessel, hub, flow, network, field, atmosphere, resource, exchange`

Figure 6: A DRAT item with OLMo-2-32B as composer and divergent-word-tuned SmolLM-135M as proposer: four of the proposer's 20 candidate lists, the composer's ten-word answer without the proposer, and its answer given the candidates. Highlighted words are taken from the candidates; the rest are the composer's own. The judge rates each noun from 0 to 4, and a list's rating is the mean over its ten nouns; the composer rises from 1.90 alone to 3.20 with the candidates. Full prompts are given in Appendix D.

Table 23: Composer accuracy (%) alone and with PCH using divergent-word-tuned proposers on OMEGA-CONSTRAINT and OMNI-MATH-CONSTRAINT, the third mode of Table 6. Each cell gives the composer's accuracy with the best of the ten tuned proposers and its change over *alone* (green ↑: rise); bold marks a significant gain.

	OMEGA						OMNI-MATH					
	OLMo-2-32B			Qwen3-32B			OLMo-2-32B			Qwen3-32B		
Condition	alone	div.-tuned		alone	div.-tuned		alone	div.-tuned		alone	div.-tuned	
baseline	8.0	↑39.5	**47.5**	70.0	↑3.0	73.0	4.5	↑5.0	**9.5**	21.5	↑1.5	23.0
length	8.0	↑31.0	**39.0**	9.0	↑13.0	**22.0**	4.0	↑4.5	**8.5**	2.0	↑6.0	**8.0**
method	8.0	↑38.0	**46.0**	50.0	↑2.0	52.0	4.0	↑4.0	**8.0**	12.0	↑3.5	15.5
char	6.0	↑38.5	**44.5**	0.5	↑2.5	3.0	4.5	↑3.0	7.5	0.0	↑0.5	0.5
length+method	8.0	↑35.0	**43.0**	8.0	↑5.5	**13.5**	2.5	↑7.5	**10.0**	2.0	↑4.5	**6.5**
length+char	6.5	↑32.5	**39.0**	3.5	↑9.5	**13.0**	1.5	↑8.0	**9.5**	0.0	↑1.0	1.0
method+char	5.5	↑38.0	**43.5**	0.0	↑0.5	0.5	3.5	↑1.5	5.0	0.0	↑0.5	0.5
all	4.5	↑32.0	**36.5**	1.5	↑9.5	**11.0**	1.5	↑6.0	**7.5**	0.0	↑1.0	1.0

T DEVELOPMENT VERSUS SELECTION IN PCH

Table 6 reports the best of ten proposers in each cell. This section examines the composer's gains from two sides: their dependence on the choice of proposer, and the composer's contribution beyond the candidates it is given. §T.1 gives the mean over the ten proposers, the accuracy of an arbitrary choice. §T.2 works through an OMEGA item on which the composer is right although most of its candidates are wrong. §T.3 reports each proposer's own accuracy on the problems. §T.4 compares the composer's accuracy with the share of problems on which at least one of its candidates is correct, the most that selection could reach. Two checks close the section. First, the candidate prompt attributes the candidates to a smaller model in every condition, and we confirm that this wording does not affect the self-proposal baseline. Second, the three smallest proposers are fully fine-tuned on DRAT, and we confirm that their mathematical ability, minimal at their size, stays near zero before and after fine-tuning.

T.1 AN ARBITRARY PROPOSER

Table 25 repeats Tables 6 and 23 with the mean over the ten proposers in place of the highest: the accuracy of an arbitrary choice. The mean gains significantly in 50 of the 96 cells, against 62 for the highest. For OLMo-2-32B on OMEGA it gains in all 24, by $+10.7$ to $+20.1$ points over a

Table 24: Composer judge rating, alone and with PCH on MacGyver and CreativityBench, on the original problems (baseline) and at k=1 to 4 swaps. The judge rates each response on three rubric dimensions from 1 to 5 (MacGyver: tool use, step quality, goal achievement; CreativityBench: tool selection, method quality, constraint respect); the item score is the mean of the three, rescaled to 0–100 as $100\,(\bar{r}-1)/4$, so a rating of 1 maps to 0 and 5 to 100. *alone*: the composer with no proposer; each mode cell (plain, SSoT, div.-tuned) gives the composer's rating with the best of the ten proposers and its change over *alone* (green ↑: rise, red ↓: drop). Bold marks a significant gain.

		OLMo-2-32B						Qwen3-32B					
Benchmark	Condition	alone	plain		SSoT		div.-tuned	alone	plain		SSoT		div.-tuned
MacGyver	baseline	46.4	↑4.4	50.8	↑8.0	**54.4**	↑7.4	**53.8**	70.5	↑7.7	**78.2**	↑8.1	**78.6**	↑8.0	**78.5**
	k=1	46.2	↑7.3	**53.5**	↑5.0	**51.2**	↑5.2	**51.4**	68.8	↑10.8	**79.6**	↑9.5	**78.3**	↑11.2	**80.0**
	k=2	43.9	↑7.7	**51.6**	↑9.4	**53.3**	↑6.6	**50.5**	68.3	↑10.7	**79.0**	↑9.9	**78.2**	↑10.8	**79.1**
	k=3	46.3	↑6.6	**52.9**	↑2.8	49.1	↑3.9	**50.2**	70.9	↑6.2	**77.1**	↑7.8	**78.7**	↑9.0	**79.9**
	k=4	41.2	↑7.2	**48.4**	↑8.3	**49.5**	↑7.8	**49.0**	70.3	↑6.0	**76.3**	↑6.1	**76.4**	↑5.5	**75.8**
CreativityBench	baseline	45.0	↑10.5	**55.5**	↑12.9	**57.9**	↑7.0	**52.0**	66.6	↑0.9	67.5	↓0.1	66.5	↑1.6	68.2
	k=1	45.3	↑11.0	**56.3**	↑12.5	**57.8**	↑6.9	**52.2**	65.2	↑2.0	67.2	↑1.6	66.8	↑4.1	**69.3**
	k=2	46.7	↑11.8	**58.5**	↑8.7	**55.4**	↑6.3	**53.0**	63.5	↑3.5	**67.0**	↑3.0	66.5	↑5.0	**68.5**
	k=3	45.1	↑13.5	**58.6**	↑12.0	**57.1**	↑7.7	**52.8**	62.7	↑5.2	**67.9**	↑4.6	**67.3**	↑5.0	**67.7**
	k=4	45.7	↑11.4	**57.1**	↑9.0	**54.7**	↑6.0	**51.7**	62.3	↑2.1	64.4	↑3.1	65.4	↑2.8	65.1

Table 25: Composer accuracy (%) alone and with PCH, averaged over the ten proposers, on OMEGA-CONSTRAINT and OMNI-MATH-CONSTRAINT. *alone*: the composer with no proposer; each mode cell gives the composer's mean accuracy over the ten proposers and its change over *alone* (green ↑: rise, red ↓: drop; a tie is unmarked). Bold marks a significant gain; since the test uses each problem's share of proposers, cells with equal means can differ in significance. Companion to Table 6, which reports the highest accuracy over the same ten proposers rather than the mean.

		OLMo-2-32B						Qwen3-32B					
Benchmark	Condition	alone	plain		SSoT		div.-tuned	alone	plain		SSoT		div.-tuned
OMEGA	baseline	8.0	↑19.8	**27.8**	↑20.1	**28.1**	↑16.4	**24.4**	70.0		70.0	↓0.2	69.8	↓1.0	69.0
	length	8.0	↑13.3	**21.3**	↑14.9	**22.9**	↑10.7	**18.7**	9.0	↑3.2	**12.2**	↑4.5	**13.5**	↑1.2	10.2
	method	8.0	↑17.1	**25.1**	↑16.9	**24.9**	↑16.3	**24.3**	50.0	↓0.6	49.4	↓2.2	47.8	↓2.5	47.5
	char	6.0	↑19.6	**25.6**	↑20.1	**26.1**	↑16.1	**22.1**	0.5	↑0.4	0.9	↑0.9	**1.4**	↑0.7	1.2
	length+method	8.0	↑14.9	**22.9**	↑17.2	**25.2**	↑13.0	**21.0**	8.0	↓1.2	6.8	↓1.2	6.8	↑0.1	8.1
	length+char	6.5	↑14.9	**21.4**	↑16.0	**22.5**	↑11.5	**18.0**	3.5	↑4.3	**7.8**	↑2.9	**6.5**	↑4.2	**7.8**
	method+char	5.5	↑19.6	**25.1**	↑19.9	**25.4**	↑16.0	**21.5**	0.0	↑0.1	0.1	↑0.2	0.2	↑0.1	0.1
	all	4.5	↑17.2	**21.7**	↑15.5	**20.0**	↑13.6	**18.1**	1.5	↑5.5	**7.0**	↑4.6	**6.1**	↑6.0	**7.5**
OMNI-MATH	baseline	4.5	↑1.5	6.0	↑0.8	5.3	↑1.2	5.8	21.5	↓0.6	20.9	↓1.2	20.3	↓1.4	20.1
	length	4.0	↑2.0	6.0	↑2.8	**6.8**	↑1.0	5.0	2.0	↑4.0	**6.0**	↑4.7	**6.7**	↑4.2	**6.2**
	method	4.0	↑0.6	4.6	↑0.7	4.7	↑0.2	4.2	12.0	↑0.5	12.5	↑0.2	12.2	↑0.5	12.5
	char	4.5	↑0.1	4.6	↑0.8	5.3	↑0.3	4.8	0.0	↑0.1	0.1	↑0.1	0.1	↑0.1	0.1
	length+method	2.5	↑3.2	**5.7**	↑2.8	**5.3**	↑2.7	**5.2**	2.0	↑2.2	**4.2**	↑2.6	**4.6**	↑2.5	**4.5**
	length+char	1.5	↑3.9	**5.4**	↑4.0	**5.5**	↑3.5	**5.0**	0.0	↑0.5	0.5	↑0.5	**0.5**	↑0.5	0.5
	method+char	3.5	↑0.3	3.8	↑0.6	4.1	↓0.3	3.2	0.0	↑0.1	0.1	↑0.1	0.1	↑0.1	0.1
	all	1.5	↑2.9	**4.4**	↑3.5	**5.0**	↑3.0	**4.5**	0.0	↑0.2	0.2	↑0.2	0.2	↑0.2	0.2

composer-alone accuracy of 4.5 to 8.0: an arbitrary proposer gives the composer three to five times what it reaches by itself. Qwen3-32B, already strong without a constraint, gains significantly from an arbitrary proposer in 9 of its 24 OMEGA cells, all under constraint. PCH supplies what a weak composer lacks whichever proposer is used, while a strong composer depends on which candidates it is given.

T.2 A WORKED OMEGA EXAMPLE

Figure 7 shows an OMEGA item where the composer returns the correct answer even though most of its candidates do not contain it. Alone, OLMo-2-32B returns `\boxed{24}` and nothing else, which is not even a factor of 30,688. Given five Phi-4-mini candidates it answers correctly, and not

Problem: What is the greatest common factor of 30688 and 168? **Gold:** 56

Composer alone (OLMo-2-32B): returns $\boxed{24}$ and nothing else, not even a factor of 30688.

Proposer candidates (Phi-4-mini; answers $8/56/8/56/8$, the majority wrong):
Candidate 1: $30688 = 168 \times 182 + 64$ (product miscomputed as 30624) $\Rightarrow$ $\gcd(168, 64) \rightarrow \boxed{8}$
Candidate 2: $30688 = 168 \times 182 + 112$ $\Rightarrow$ $\gcd(168, 112) = \gcd(112, 56) \rightarrow \boxed{56}$
Candidate 3: $30688 = 168 \times 182 + 64$ $\Rightarrow$ $\gcd(168, 64) \rightarrow \boxed{8}$
Candidate 4: $30688 = 168 \times 182 + 112$ $\Rightarrow$ $\boxed{56}$
Candidate 5: $168 \times 183 = 30744$, remainder taken as 64 $\Rightarrow$ $\boxed{8}$

Composer + candidates: recomputes $168 \times 182 = 30576$, remainder $30688 - 30576 = 112$, then $\gcd(168, 112) = 56$, and returns $\boxed{56}$.

Figure 7: An OMEGA item, the greatest common factor of 30,688 and 168 (gold 56), with OLMo-2-32B as composer and Phi-4-mini as proposer: the composer's answer alone, the five candidates with the answer each reaches, and the composer's answer given the candidates. Three of the five candidates miscompute the first Euclidean division and end at 8; the composer redoes the division itself and returns the minority answer, 56.

by voting: three of the five miscompute the first Euclidean division and end at 8, while two perform the division correctly and reach 56. The composer redoes the division, follows the correct remainder chain, and returns the minority answer 56. Such cases are common: over all OMEGA conditions, proposers, and modes, OLMo-2-32B answers correctly 11,138 times; in 2,658 of these one or two of its five candidates are correct, and in 1,194 no candidate is.

T.3 PROPOSERS AS SOLVERS

Table 26 reports each proposer's own accuracy on the problems, pooled over the eight conditions. Candidates are brainstorming material for the composer, so they are not penalized for violating a constraint here. Table 27 asks a different question, whether the composer outperforms the best of its own candidates, and there a candidate is judged as an answer, so a violation counts as wrong. The small proposers, as their size predicts, are near zero on the math benchmarks (at pass@1, the two GPT-2 variants stay at 0.0, SmolLM-135M and OLMo-2-1B reach at most 0.3 and 2.5), and their candidates bring the composer little there. On DRAT the order reverses, and the smallest proposers are among the best (Table 5): word associations are within reach of a small model, while solving a competition problem is not. For the larger proposers, the gap between pass@1 and pass@5 is wide: Phi-4-mini-reasoning reaches 48.6 at pass@1 on OMEGA against 67.8 at pass@5, nearly 20 points of headroom sitting in its five candidates. PCH turns this headroom into final answers: with its best proposer, OLMo-2-32B gains 45.0 to 66.0 points across the OMEGA conditions in the plain and SSoT modes (Table 6), reaching above every proposer's pass@1 and, under every OMEGA constraint, above the share of problems with a correct compliant candidate (Table 27). Qwen3-32B alone is correct on 53.2% of the OMEGA problems when compliance is not counted, higher than every proposer's pass@1, so its gains are smaller, yet it improves in every OMEGA and OMNI-MATH condition (Table 6); on OMEGA it rises from 3.5 to 22.5 under the length and character constraints and from 1.5 to 16.0 under all three. Fine-tuning on DRAT moves the proposers' own pass@1 by less than 5 points, with two exceptions, both on OMEGA: Phi-4-mini gains 10.2 and Phi-4-mini-reasoning drops 44.5.

Winning proposers. Figure 8 marks the proposer with which the composer reaches its highest accuracy in each cell of Tables 6 and 23. A single proposer wins in most cells; where two or more proposers tie, each is marked. On OMEGA, Phi-4-mini-reasoning, the strongest separate solver in plain and SSoT mode (Table 26), wins most plain and SSoT cells for both composers and every one for OLMo-2-32B; divergent-word tuning lowers its own accuracy from 48.6 to 4.1, and the tuned wins move to other proposers. Qwen3-32B often wins with itself as proposer, most of all on OMNI-MATH, where it solves more problems on its own than any other proposer.

Table 26: Accuracy (%) of each proposer solving the problems on its own, pooled over the eight conditions of Table 6; a constraint violation is not counted as wrong here. Plain, SSoT, and divergent-word-tuned are the three proposer modes; p@1 (pass@1) is the share of the five candidates per problem that are correct and p@5 (pass@5) the share of problems with at least one correct candidate. The *itself* rows use a composer as its own proposer; the composer-alone rows are single greedy samples, so p@5 is undefined for them. At p@1, even the strongest proposer stays below Qwen3-32B alone, with nearly 20 points between its p@1 and p@5.

	OMEGA						OMNI-MATH					
	plain		SSoT		div.-tuned		plain		SSoT		div.-tuned	
Proposer	p@1	p@5	p@1	p@5	p@1	p@5	p@1	p@5	p@1	p@5	p@1	p@5
GPT-2	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
GPT-2 genre-story	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SmolLM-135M	0.3	1.5	0.3	1.5	0.2	0.8	0.0	0.2	0.0	0.2	0.1	0.2
OLMo-2-1B	2.3	6.9	2.5	7.5	2.5	7.5	1.3	4.5	1.4	4.3	0.9	3.1
Qwen3-1.7B	36.1	48.6	32.4	46.3	35.0	50.6	5.7	10.9	5.8	11.0	5.7	10.3
SmolLM3-3B	33.6	53.8	33.7	55.1	34.8	54.8	7.2	14.7	6.8	15.3	7.5	15.2
SmolLM3-3B (thinking)	36.6	58.2	35.8	58.5	32.0	54.2	7.3	15.5	6.8	15.9	5.6	13.1
Phi-4-mini	22.5	38.7	22.7	41.5	32.7	55.3	6.0	13.6	4.9	11.9	6.4	12.7
Phi-4-mini-reasoning	48.6	66.7	48.6	67.8	4.1	12.4	4.5	8.6	4.9	9.3	1.4	4.8
itself (OLMo-2-32B)	13.9	28.8	12.6	28.4	10.8	24.3	3.4	9.1	3.7	9.6	3.1	7.6
itself (Qwen3-32B)	47.1	65.1	47.8	64.9	47.4	65.8	9.2	17.4	10.0	18.9	9.2	17.4
OLMo-2-32B alone			6.9						3.8			
Qwen3-32B alone			53.2						16.7			

T.4 CANDIDATES VERSUS COMPOSER

Table 27 tests whether the composer's accuracy could be explained by selection alone. For the proposer behind each composer's best cell, the table reports three numbers per condition: the share of candidates that are correct and comply with the constraint (cand. p@1); the share of problems on which at least one candidate does (cand. p@5), which is the highest accuracy that selection could reach; and the composer's accuracy with those candidates. In 23 of the 28 constrained cells the composer's accuracy exceeds cand. p@5, so on some problems it produces a correct compliant answer that no candidate contained: development rather than selection. The margins are often wide: on OMEGA under all three constraints, OLMo-2-32B goes from 4.5 alone to 63.0 with candidates, although only 40.5% of problems have a correct compliant candidate, and under the character constraint from 6.0 to 60.5, although only 2.0% do. The remaining cells are OLMo-2-32B's two baselines, where no constraint applies and at least one candidate is correct on more problems than the composer solves (63.5 against 59.0 on OMEGA, 12.0 against 9.5 on OMNI-MATH), and five constrained cells where the composer's accuracy does not exceed cand. p@5: OLMo-2-32B under the method constraint on OMNI-MATH, Qwen3-32B under length+character, method+character, and all three constraints on OMEGA, and Qwen3-32B under length+method on OMNI-MATH. Three of the five are within one point of cand. p@5; in the other two, Qwen3-32B under length+character and under all three constraints on OMEGA, the composer is well below the candidates. The complementary question, whether the composer's own samples would reach the same accuracy at the same budget, is addressed in Appendix V.

Prompt attribution in the self-proposal condition. The composer prompt is the same in every condition, so that conditions differ in their candidates alone; a prompt written for one condition would introduce the difference it is meant to remove. The candidate block attributes the candidates to a smaller model, "a small brainstorming model produced these candidate answers" on DRAT and "a smaller model produced these draft attempts" on the transfer benchmarks, and adds a caution about their quality. The block is written so that the composer builds on the candidates rather than copying them, and it is left unchanged in the self-proposal condition because a prompt that identifies the candidates as the composer's own would encourage it to restate them, which is what the condition

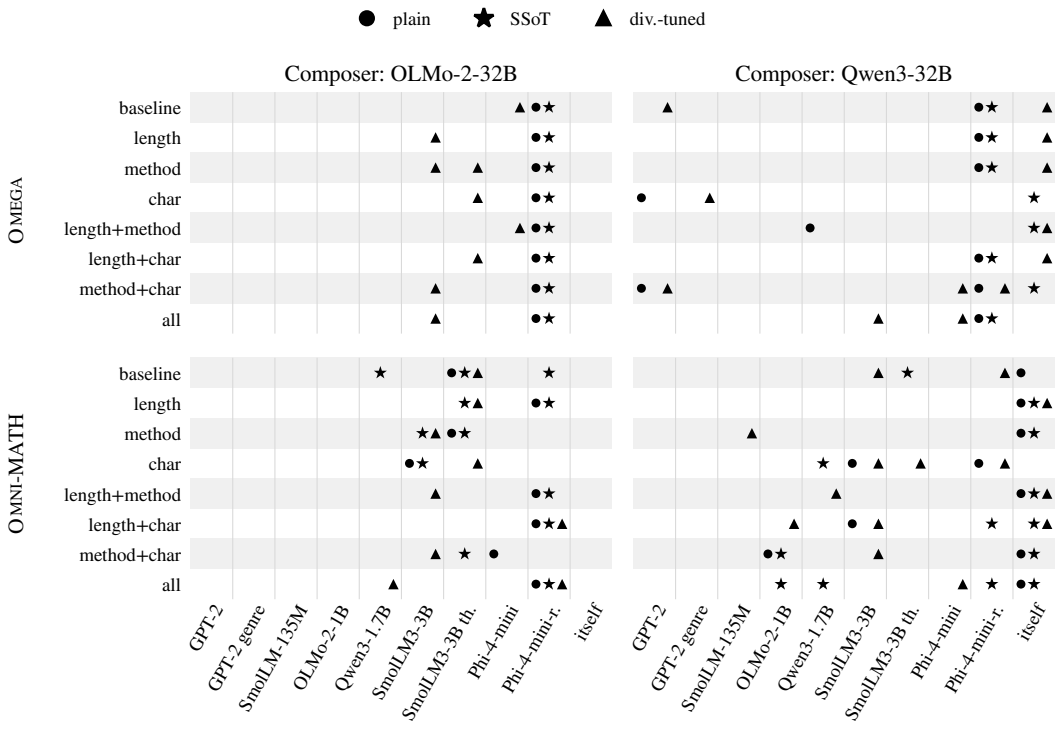


Figure 8: Winning proposer in each cell of Tables 6 and 23. For each composer, benchmark, and condition, a mark under a proposer means the composer reaches its highest accuracy with that proposer in one mode (• plain, ★ SSoT, ▲ divergent-word-tuned); when several proposers give the composer the same highest accuracy, each of them is marked. A violation of the constraint counts as wrong. *itself*: the composer as its own proposer; th.: thinking mode; r.: the reasoning variant.

is meant to exclude. The caution applies in every condition alike; the attribution is the one statement that is untrue when the composer is its own proposer, so we test its effect. We rerun the self-proposal condition in plain mode for OLMo-2-32B on all eight OMEGA conditions with the same candidates and the same decoding, replacing the attribution with "these draft attempts were produced by a model". Accuracy changes by -0.56 points pooled over the 1,600 item–condition pairs ($p = .15$, paired permutation test) and by at most 2.5 points in either direction in any single condition (Table 28). On DRAT we make the same replacement for both composers in plain mode, "these candidate answers were produced by a model", again with the same candidates and decoding, and score both runs in one judging run: the judge rating changes by 0.00 for OLMo-2-32B ($p = .99$) and by $+0.07$ for Qwen3-32B ($p = .47$). The attribution does not change how the composer uses its own candidates on either task.

Candidates from the fully fine-tuned proposers on mathematics. Three proposers are fully fine-tuned rather than through an adapter: the two GPT-2 variants and SmolLM-135M. Figure 9 shows their candidates on an OMEGA problem: the prompt restated, then the vocabulary of DRAT, with no arithmetic attempted. Their accuracy stays near zero before and after fine-tuning, so this is a property of their scale rather than of the fine-tuning: on OMEGA at pass@1, the two GPT-2 variants are at 0.0 in both the plain and the divergent-word-tuned mode, and SmolLM-135M at 0.3 and 0.2 (Table 26). At 124M and 135M parameters these models do not solve competition problems in any mode, and the composer takes no content from them, as in the figure. The mathematics gains of PCH come from the proposers that can attempt the problems.

Table 27: Candidates versus composer, with each composer's best proposer from Table 6: the share of candidates that are correct and comply with the constraint (cand. p@1), the share of problems with at least one such candidate (cand. p@5), and the composer's accuracy with those candidates. Bold marks a cell where the composer's accuracy exceeds cand. p@5, the most that selecting a candidate could reach.

		OLMo-2-32B			Qwen3-32B		
Benchmark	Condition	cand. p@1	cand. p@5	composer	cand. p@1	cand. p@5	composer
OMEGA	baseline	48.4	63.5	59.0	48.3	63.5	**85.0**
	length	6.1	21.5	**74.0**	6.1	21.5	**28.0**
	method	22.7	38.5	**54.5**	22.2	38.0	**59.0**
	char	0.4	2.0	**60.5**	1.1	3.5	**4.0**
	length+method	5.7	18.5	**67.5**	4.9	12.5	**13.5**
	length+char	26.9	53.5	**71.0**	29.2	56.0	22.5
	method+char	0.3	1.5	**54.5**	0.3	1.0	1.0
	all	18.6	40.5	**63.0**	23.5	43.5	16.0
OMNI-MATH	baseline	6.0	12.0	9.5	6.3	11.0	**26.5**
	length	2.0	5.0	**15.5**	4.3	9.5	**10.5**
	method	3.5	9.0	8.0	3.1	7.0	**16.5**
	char	0.0	0.0	**9.0**	0.0	0.0	**0.5**
	length+method	0.3	1.5	**14.0**	4.0	8.5	8.0
	length+char	1.5	5.0	**16.0**	0.0	0.0	**1.5**
	method+char	0.0	0.0	**6.0**	0.0	0.0	**0.5**
	all	1.5	5.0	**14.0**	0.0	0.0	**1.0**

Table 28: Prompt attribution in the self-proposal condition, where the composer receives its own plain-mode candidates. Top: OLMo-2-32B on OMEGA-CONSTRAINT, accuracy (%) with a constraint violation counted as wrong; *original*: "a smaller model produced these draft attempts", *neutral*: "these draft attempts were produced by a model". Bottom: both composers on DRAT, judge rating (0–4); *original*: "a small brainstorming model produced these candidate answers", *neutral*: "these candidate answers were produced by a model". Candidates and decoding are the same; the attribution sentence alone differs. p from the paired permutation test.

	original	neutral	diff	p
baseline	27.5	28.5	+1.0	.50
length	7.5	5.5	−2.0	.13
method	24.5	23.5	−1.0	.63
char	21.0	20.0	−1.0	.50
length+method	12.0	9.5	−2.5	.13
length+char	4.5	4.5	0.0	1.00
method+char	20.5	22.5	+2.0	.12
all	10.5	9.5	−1.0	.62
OLMo-2-32B	2.37	2.38	0.00	.99
Qwen3-32B	1.99	2.06	+0.07	.47

U SELF-REFINE AS A BASELINE

Self-Refine (Madaan et al., 2023) uses one model in three roles: the composer writes an answer, critiques it, and revises it. Each round therefore takes two calls to the composer, one to critique and one to revise. Following Madaan et al. (2023), we run up to three rounds, stopping early when the model reports that no further refinement is needed, so an item takes at most six calls after the first draft; OLMo-2-32B uses 4.3 on average and Qwen3-32B all six. The first draft is the composer's *alone* answer of Table 5, so Self-Refine and PCH are measured from the same baseline. Table 29 gives the comparison. Self-Refine does not improve OLMo-2-32B (-0.05) and improves Qwen3-32B by $+0.27$, against $+0.83$ and $+0.38$ for PCH, significant for both composers. Self-Refine also uses more composer calls than PCH: 5.3 and 7.0 per item in total, against one for PCH, which adds

Problem: What is the greatest common divisor of 558081 and 9? **Gold:** 9

Candidates from GPT-2 (124M), fine-tuned on DRAT (2 of 5):

```
Let's look at the formula for the greatest common divisor of 558081 and 9.  Present
the solution in LaTex format:  \boxed{Your answer} Full solution, ending in \boxed{},
respecting every constraint:
Your solution, ending in \boxed{}, respecting every constraint, is a unique solution with
different anchors, meaning, different approaches, different sizes, different approaches,
different outcomes
```

Composer alone: `\boxed{9}` *correct*

Composer + candidates: `\boxed{9}` *correct*

Figure 9: An unconstrained OMEGA item with OLMo-2-32B as composer and GPT-2 (124M), fully fine-tuned on DRAT, as proposer. The candidates restate the prompt and continue in the vocabulary of DRAT. The composer answers correctly with and without the candidates.

Table 29: Self-Refine against PCH on the 60 DRAT items, judge rating (0–4). *alone* and *PCH* are the values of Table 5; PCH uses the divergent-word-tuned SmolLM-135M proposer. Self-Refine adds on average 4.3 (OLMo-2-32B) and 6.0 (Qwen3-32B) composer calls per item to the first answer; PCH adds no composer call, since its one composer call replaces that of the composer alone, though that call takes the candidates as input and costs more (Appendix V); it adds 20 calls of the 135M proposer. Bold marks a significant gain over *alone*; p from the paired permutation test.

Composer	alone	Self-Refine	p	PCH	p
OLMo-2-32B	2.34	2.30 (-0.05)	.63	**3.17** ($+0.83$)	$< .0001$
Qwen3-32B	2.10	**2.36** ($+0.27$)	.017	**2.48** ($+0.38$)	.003

20 calls of a 135M proposer instead. PCH's gain therefore comes from candidates written by a separate model, and further passes over the composer's own answer are no substitute.

V COMPUTE-MATCHED BASELINES

Table 30 gives the inference cost of PCH relative to the composer alone, counting $2P$ FLOPs per token for a model with P parameters. The proposer accounts for 10–12% of PCH's added cost over the composer alone. The rest is the composer, almost all of it prefill of the candidate block, since gpt-4.1 and Qwen3-32B generate fewer tokens in PCH than alone. PCH's budget equals 4.8 to 13.5 composer samples; the exception is OLMo-2-32B, which answers in about seven tokens when run alone, so the same budget would buy 266 of its samples on OMEGA and 198 on OMNI-MATH. Measured on 50 OMEGA-CONSTRAINT items with Qwen3-32B on one A100 with batched decoding, the composer alone takes 1,720 s, PCH 1,739 s ($1.011\times$), and five samples from the composer alone 2,168 s ($1.260\times$): PCH costs more FLOPs than five samples but less time, because its added cost is prefill rather than generation. We draw k samples from the composer alone at $T = 0.7$. *Self-consistency* takes the majority answer among the compliant samples. *pass@k* credits an item if any of the k samples is correct; it is therefore an upper bound on any procedure that returns one of the k candidates, including search over them. Qwen3-32B is reported at $k = 5$ ($3.9\times$, against PCH's $4.8\times$) and OLMo-2-32B at $k = 20$. More samples barely help self-consistency: with OLMo-2-32B it moves by at most 2 points in any condition between $k = 3$ and $k = 20$, with no consistent direction. PCH improves over the greedy composer in every condition, and is above pass@k at the matched budget in all eight conditions with OLMo-2-32B and five of the eight with Qwen3-32B (Table 31, bold). Averaged over the ten plain proposers rather than taking the best, PCH gives 21.7 for OLMo-2-32B under all three constraints, still above pass@20's 17.5. Sampling does not reduce constraint violations either: over the Qwen3-32B conditions the compliant share of samples is 9.4% against 8.5% for the greedy pass under all three constraints, 1.1% against 1.0% under the character constraint, and 36.4% against 34.0% under the length constraint.

Table 30: Inference cost of PCH relative to the composer alone, with KV caching and $2P$ FLOPs per token for prefill and decode alike, P the parameter count. Q: problem tokens. S: candidate-block tokens. D_a, D_h: composer output tokens alone and in PCH. *FLOPs* $\times$: PCH cost over composer alone. *proposer*: share of the added cost spent in the proposer. *prefill*: share of the added cost spent on prefill (above 100% when the composer generates fewer tokens in PCH than alone). k^*: number of composer samples, sharing one prefill of the problem, whose cost equals PCH.

Benchmark	Composer	Q	S	D_a	D_h	FLOPs $\times$	proposer	prefill	k^*
OMEGA	gpt-4.1	83	1989	540	499	6.23	11%	91%	13.5
	OLMo-2-32B	92	1578	7	39	21.26	11%	87%	265.8
	Qwen3-32B	93	1666	473	398	4.80	12%	99%	6.7
OMNI-MATH	gpt-4.1	117	3642	1251	1139	4.40	10%	93%	5.1
	OLMo-2-32B	125	2566	84	122	19.78	10%	92%	198.0
	Qwen3-32B	125	2695	1002	802	4.09	11%	102%	4.8

Table 31: Compute-matched sampling baselines against PCH on OMEGA-CONSTRAINT, accuracy (%). *alone*: greedy, one pass. *SC@*k: majority answer among the compliant samples of k draws at $T = 0.7$. *pass@*k: any of the k is correct and compliant. *PCH*: plain mode, with the best of the ten proposers; bold marks PCH above pass@k. The sampling budget is $k = 20$ for OLMo-2-32B ($5.1\times$ a greedy pass, against $21.3\times$ for PCH) and $k = 5$ for Qwen3-32B ($3.9\times$, against PCH's $4.8\times$).

	OLMo-2-32B ($k = 20$)				Qwen3-32B ($k = 5$)			
Condition	alone	SC@k	pass@k	PCH	alone	SC@k	pass@k	PCH
baseline	8.0	9.0	24.0	**58.0**	70.0	71.5	85.5	85.0
length	8.0	8.0	22.0	**72.5**	9.0	12.0	12.5	**22.0**
method	8.0	9.5	28.0	**53.0**	50.0	55.0	65.0	59.0
char	6.0	6.5	19.5	**59.0**	0.5	2.5	2.5	**3.5**
length+method	8.0	9.5	19.5	**67.0**	8.0	15.5	16.5	10.5
length+char	6.5	7.5	16.5	**67.0**	3.5	7.0	7.5	**22.5**
method+char	5.5	6.0	19.0	**51.0**	0.0	0.0	0.0	**0.5**
all	4.5	5.5	17.5	**59.0**	1.5	5.5	5.5	**16.0**

W COMPUTE AND SOFTWARE ENVIRONMENT

All open-weight models were served and fine-tuned on an HPC cluster with NVIDIA A100 (80 GB), RTX PRO 6000, and A40 GPUs, using the software listed in §W.2; the API models were queried through their providers.

W.1 COMPUTE RESOURCES

The reported experiments took about 5,200 GPU-hours, nearly all in inference: roughly 3,300 for PCH (about 950 for candidate generation and 2,300 for composition, over DRAT and the 1,560-run transfer grid), 1,500 for LLM judging and compliance checks, and 350 for evaluating the locally served models on the remaining experiments; fine-tuning the proposers (Appendix J) took about 10 GPU-hours in total. The API models (gpt-4.1, gpt-5.5, gpt-5.4-mini, claude-sonnet-4.6, and kimi-k2-thinking) cost roughly $600–700 in total.

W.2 LIBRARIES

Table 32 lists the software used.

Table 32: Software used, with version, license, and GitHub repository.

Library	Version	License	Repository
PyTorch	2.12.0	BSD-3-Clause	pytorch/pytorch
Transformers	5.9.0	Apache-2.0	huggingface/transformers
PEFT	0.20.0	Apache-2.0	huggingface/peft
Accelerate	1.13.0	Apache-2.0	huggingface/accelerate
Datasets	4.8.5	Apache-2.0	huggingface/datasets
sentence-transformers	5.5.1	Apache-2.0	UKPLab/sentence-transformers
NumPy	2.4.6	BSD-3-Clause	numpy/numpy
SymPy	1.14.0	BSD-3-Clause	sympy/sympy
SciPy	1.17.1	BSD-3-Clause	scipy/scipy
pandas	3.0.3	BSD-3-Clause	pandas-dev/pandas
Matplotlib	3.11.1	PSF-based	matplotlib/matplotlib
NLTK	3.9.4	Apache-2.0	nltk/nltk
wordfreq	3.1.1	Apache-2.0	rspeer/wordfreq
OpenAI SDK	2.40.0	Apache-2.0	openai/openai-python
Anthropic SDK	0.125.0	MIT	anthropics/anthropic-sdk-python