CounterPlay: Evaluating Counterfactual World Simulation in LLMs Through Games

Anonymous ACL submission

Abstract

Counterfactual thinking, or the ability to construct coherent alternative worlds and meaningfully simulate them, has long been considered essential to human creativity. In this work, we study the ability of modern LLMs to correctly simulate counterfactuals by introducing a benchmark of verifiable 'counterfactual worlds' derived from canonical games. For each game, we alter a governing rule of the game or a property of the closed world in the game, to construct a novel world where successful gameplay requires recomputing consequences of actions in the new world. Our benchmark, CounterPlay, spans 8 games, 22 counterfactual variants, and 2,365 puzzles. Each solution is evaluated by a deterministic game-specific verifier, making the benchmark automatically checkable. Across 15 LLMs, we find that counterfactual variants remain challenging even when models can solve the corresponding standard games: changing one rule reduces accuracy by a median of 15 percentage points. Error analysis shows that models sometimes produce solutions that remain valid for the standard game but violate the counterfactual rules. Some easier variants can be handled by translating the task back into a familiar baseline form, whereas counterfactual games are hardest when the rule change invalidates the default solution strategy or substantially lengthens the solution. Taken together, our results show that there is still considerable room for progress in enabling models to reason about alternative worlds, a capability central to creativity and intelligence.[1]

1 Introduction

"The giant number of brain cells interposed between stimulus and action is a critical contributor to the massive creativity of our species. It is what allows us to consider possibilities beyond what is right in front of us. And that's a large part of the magic of human brains: we relentlessly simulate what-ifs."

— David Eagleman and Anthony Brandt, The Runaway Species: How Human Creativity Remakes the World

Humans possess the remarkable ability to mentally simulate alternative versions of reality, inferring how outcomes might have differed had a specific condition been changed, without ever having experienced those alternatives firsthand. This ability, reasoning about counterfactuals, is a hallmark of human cognition (Pearl and Mackenzie, 2018; Byrne, 2016) and is considered to be key to enabling human creativity and imagination (Markman et al., 2007; Hoeck et al., 2015). For LLMs, it is also diagnostic of generalization capabilities: a model that has memorized the standard form of a task may fail once the rules are altered, even when the surface form is unchanged (Wu et al., 2024).

We evaluate LLMs on their ability to reliably simulate counterfactual scenarios. To construct a controlled evaluation where it is possible to algorithmically verify correctness of these simulations, we contribute a benchmark of 'counterfactual worlds' that are derived from canonical games. Each counterfactual world is constructed by modifying a rule or environmental property of the original game while preserving its surface structure. We evaluate the ability of the LLMs to execute successful gameplay in these counterfactual worlds, which requires adapting their actions by reasoning about the counterfactual world, rather than relying on memorized strategies from the original game. Familiar games (Sudoku, Tower of Hanoi) carry strong priors that the model can exploit by recitation (Yan et al., 2025). The counterfactual variants are unlikely to appear verbatim in pretraining, yet their solutions remain algorithmically verifiable.

[1]The resource available at: https://anonymous.4open.science/r/counterplay-A37E

Game / Variant	Descriptions of the baseline games and their counterfactual (CF) variants
1. Sudoku	9×9 *grid. The digits 1–9 must appear exactly once in every row, column, and* 3×3 *box.*
CF1 double-8	The digit 9 is removed and the digit 8 appears twice in every row, column, and box.
CF2 shifted boxes	The nine 3×3 boxes are shifted by one cell with toroidal wrap-around (rows and columns unchanged).
CF3 11-19	The alphabet is relabeled from $\{1,\ldots,9\}$ to $\{11,\ldots,19\}$, so every symbol is two digits.
CF4 hex 16×16	The grid grows to 16×16 with 4×4 boxes and a hexadecimal alphabet $\{0,\ldots,9,\mathrm{A},\ldots,\mathrm{F}\}$.
2. Cryptarithmetic	*base-10 addition* $W_1 + W_2 = R$*, where each letter stands for a distinct digit.*
CF1 base-9 add	The same equation shape, but every column carries at 9 instead of 10 and the digit 9 is removed.
CF2 base-9 sub	Subtraction $W_1 - W_2 = R$ in base 9, where borrowing happens at 9 instead of at 10.
CF3 base-9 mixed	Chained $W_1 + W_2 - W_3 = R$ in base 9, mixing carry and borrow in a single computation.
3. Countdown	*six source numbers and a target. Combine a subset with* $+, -, \times, \div$ *to reach the target.*
CF1 base-9	Every numeral in the prompt and in the answer expression is written in base 9.
CF2 no-subtraction	The minus operator is forbidden. Only $+, \times, \div$ are allowed.
CF3 use-all	All six source numbers must appear in the expression, each used exactly once.
CF4 multi-target	The six numbers are split into two fixed groups of three, and each group must reach its own target.
4. Game-24	*combine four given numbers with* $+, -, \times, \div$ *and parentheses to reach* 24*, each used once.*
CF1 base-9	The four operands and the target are presented in base 9 (so the target "24" decimal becomes "26").
CF2 each-op-once	No operator may be repeated. At most one of each $+, -, \times, \div$ in the expression.
CF3 prime target	The target is a prime other than 24, chosen so that 24 itself is unreachable from the given multiset.
5. Water-Jugs	*two jugs of fixed capacity. Reach a target volume using fill, empty, and lossless pour moves.*
CF1 GCD-breaker	Pours incur a 1-unit evaporation tax. The target lies outside the standard $\gcd(A, B)$-lattice.
CF2 entropy trap	Pours incur a 1-unit evaporation tax. The optimal first move under classic rules fails under here.
6. Tower-of-Hanoi	*3 pegs and* n *disks. Move the stack from peg* A *to* C*, never placing a larger disk on a smaller.*
CF1 four pegs	A fourth peg is available, allowing the Frame–Stewart shortcut that beats $2^n - 1$ moves.
CF2 adjacent	The three pegs are laid in a line, and only moves between physically adjacent pegs are legal.
7. Checker-Jumping	N *red frogs, 1 empty cell,* N *blue frogs. Slide 1 cell or jump over 1 opposite frog. Swap the groups.*
CF1 asymmetric	N red frogs face $N + k$ blue frogs (with $k \in \{1, 3, 5\}$), breaking the left–right symmetry.
CF2 jump-two	In addition to the standard jump, a frog may also jump over two consecutive opposite frogs.
CF3 two-empty	The board has two empty cells between the color groups instead of one.
8. River-Crossing	*the farmer ferries Wolf, Goat, Cabbage across. Boat holds* F *+ 1.* $W{+}G$ *and* $G{+}C$ *are unsafe alone.*
CF1 neutral	An extra traveller N is added who has no predator/prey constraints with anyone.

Table 1: The games and variants. One worked example per variant appears in Figures 1, 2, 3, 4, 5, 6, 7, 8.

As baseline and counterfactuals (CF) share similar surface forms, accuracy degradations are likely due to the rule change (Mirzadeh et al., 2025).

While there has been prior interest in studying CF simulation across different task variants (Wu et al., 2024), we move beyond CF task perturbations to CF world simulations as exemplified through games. We build a benchmark of eight games and 22 CF variants (30 baseline+CF cells in total) and evaluate fifteen LLMs. **Our contributions are:** (1) A novel controlled evaluation for CF reasoning in LLMs through modified games, spanning **2,365** puzzles; (2) We evaluate 15 LLMs and show that even frontier models with strong performance on the original games suffer substantial degradation under CF modifications; (3) Through error analysis, we find multiple models still reason within the constraints of the original game and enact actions that are invalid in the CF world; (4) We analyze which CF settings are most challenging, along dimensions including the structure of the modification, the complexity of the CF world, and its proximity to the original game, and (5) Finally, we investigate how scale and reasoning can improve CF simulation capabilities, where a token-budget analysis showing that response length is largely uncorrelated with accuracy in CF scenarios.

2 Related Work

Counterfactual evaluation has become a central tool for testing whether LLMs reason or recite. Wu et al. (2024) introduces counterfactual variants of 11 tasks and reports a large, consistent accuracy drop, while Chen et al. (2026) builds CounterBench around Pearl-style simple, joint, and nested counterfactuals. Domain-specific probes apply the same principle to repeated games (Georgousis et al., 2026), redefined digits and operators (O'Brien et al., 2025), and fine-tuning on contradictory rules (Chen et al., 2025). Further arithmetic and word-problem studies investigate rule perturba-

tion, knowledge redefinition, and pedagogical counterfactuals (Li et al., 2024a; Liu et al., 2025; Balappanawar et al., 2025; Kumar et al., 2025; Sonkar et al., 2024). In contrast to robustness-oriented studies that perturb the surface while keeping rules fixed (Huang et al., 2025; Mirzadeh et al., 2025; Li et al., 2024b; Yu et al., 2025; Golikov et al., 2026), we keep the surface intact and change the rules. Moreover, in contrast to their single-domain focus, our approach encompasses eight rule-intensive puzzle domains within one unified framework.

We also evaluate counterfactual reasoning through game environments. Game-playing has long been used to probe machine cognition (Silver et al., 2017; Chollet, 2019). Recent benchmarks include Sudoku-Bench (Seely et al., 2026), Tower-of-Hanoi planning probes (Shojaee et al., 2026), and the Mathador target-number game (Kurtic et al., 2024). Closely related work also studies rule-based game reasoning with deterministic checks (Gui et al., 2025), dynamic-rule environments (van Wetten et al., 2026), and, more broadly, rule-guided reasoning in real-world domains (Zhou et al., 2025). `CounterPlay` is complementary: it compares standard and rule-altered versions of single-agent puzzles across multiple game families, enabling baseline-conditioned analyses of when and how LLMs fail to adapt to changed rules.

3 Benchmark

3.1 Design Principles

`CounterPlay` is built around paired game variants. For each game, we provide a standard **baseline** version following the textbook rules and one or more **counterfactual** (CF) variant(s) in which a single rule is explicitly changed. The task remains recognizable, but the model must solve it under the altered rule. This design allows us to measure whether models can adapt to transformed rule spaces rather than relying on default game priors.

We select games that are well-known, expressible entirely in text, and verifiable. All games are single-agent: the model produces a complete solution, and no adversarial player is modeled. The benchmark contains eight games: Sudoku, Cryptarithmetic, Countdown, Game-24, Water-Jugs, Tower of Hanoi, Checker-Jumping, and River-Crossing. The first five involve explicit numerical, algebraic, or arithmetic manipulation, while the last three are state-transition puzzles whose solutions are move sequences. We use this distinction descriptively, but our main analyses group variants by the type of CF rule change.

Each game has 1–4 CF variants chosen in a way that is easy to state in the prompt, preserves a well-defined solution space, and is automatically checkable. They cover rule alterations such as symbol or value remapping, arithmetic/base changes, constraint changes, transition-rule changes, and search-space modifications. For the first five games, where the output is a single final object, instances are generated alongside a reference solution. For expression- and move-based games, multiple valid solutions may exist. In these cases, any solution satisfying the verifier is accepted.

Overall, `CounterPlay` contains 2,365 puzzles. Each variant of Sudoku, Cryptarithmetic, Countdown, Game-24, and Water-Jugs contains 100 puzzles. Each variant of Tower of Hanoi, Checker-Jumping, and River-Crossing contains 20-30 puzzles. For these move-sequence games, we evaluate each model over 5 independent generations. Table 1 shows the games and variants, while Appendix A provides detailed descriptions and examples.

3.2 Prompting and Scoring

Each puzzle is presented in plain text with the full rule statement. Baseline and CF prompts share the same template and differ only in the rule clause. The model is asked to return a solution in a game-specific format (a Sudoku grid, a cryptarithmetic assignment, an arithmetic expression, or a sequence of moves). Outputs are parsed by deterministic game-specific parsers and scored by deterministic verifiers. Unparseable outputs count as incorrect.[2] A response is *correct* if the parsed answer satisfies all constraints under the relevant rules: for Sudoku, Cryptarithmetic, Countdown, Game-24, and Water-Jugs, the verifier checks the extracted solution directly. For Tower of Hanoi, Checker-Jumping, and River-Crossing, the move sequence is replayed step by step and must be both fully legal and end in the goal state.

4 Experimental Setup

4.1 Models

We evaluate 15 LLMs from proprietary, open-weight, small, large, and reasoning models: Claude-Opus-4.7, Claude-Sonnet-4.6, Grok-4.1-fast, Gemini-2.5-Flash, Qwen3-235B, DeepSeek-V3.2, DeepSeek R1, Llama-3.3-70B, GPT-OSS-

[2] Parsing pipeline and partial-credit metrics in Appendix B.

20B, and 120B, Gemma-3-27B, OLMo-3.1-32B, Llama-3.1-8B, Qwen3.5-9B, and OLMo-2-7B.[3]

4.2 Evaluation Protocol

Prompts contain the baseline or CF rules and output format, with no in-context demonstrations. Sudoku, Cryptarithmetic, Countdown, Game-24, and Water-Jugs are run at `temperature=0` with one generation per puzzle. Tower of Hanoi, Checker-Jumping, and River-Crossing are run at `temperature=0.7` with five generations per puzzle, following Shojaee et al. (2026). Accuracy is reported over all $5 \times |\text{puzzles}|$ runs per game variant. The maximum output length is set per game to cover the longest valid solutions, so no responses are truncated. Our primary metric is **exact-match accuracy**: a response is correct if both parseable and satisfies the game-specific verifier under the baseline or counterfactual rules.[4]

5 Results

Table 2 summarizes exact-match accuracy on `CounterPlay`. Baseline entries are exact-match accuracies, while CF entries are deltas from the corresponding baseline accuracy. Performance is highly variable across models: Opus and Sonnet lead overall (80% and 75%), followed by Gemma-3-27B, GPT-OSS-120B, DeepSeek-V3.2, and Gemini-2.5-Flash (49–56%). All other models score at or below 43%. These aggregate scores also reveal important baseline limitations. For example, only four models achieve substantial Sudoku baseline accuracy, and all non-Claude models score at most 12% on the Checker-Jumping baseline. Therefore, low CF accuracy is not always evidence of failed rule adaptation: the model may already fail the corresponding baseline game. We therefore begin by estimating CF gaps only in model–game settings with non-trivial baseline competence, and then analyze failure modes, variant properties, and the effects of scale and reasoning.

5.1 Does Baseline Competence Transfer to CF Settings?

A low CF score is uninterpretable when the model also fails the corresponding baseline game, so we condition on baseline competence: a model–game pair is included only if its baseline accuracy reaches at least 50%. For games with multiple CF variants, we average the corresponding CF accuracies before computing the gap. This yields 46 baseline-competent model–game pairs covering all eight games, with median baseline accuracy of 89%. We define the CF gap as $\Delta = \text{Acc}_{\text{CF}} - \text{Acc}_{\text{baseline}}$, where Acc_{CF} is the accuracy of the model in the CF setting, and $\text{Acc}_{\text{baseline}}$ is the accuracy of the model in the baseline setting.

Within this subset, median CF accuracy is 59%, corresponding to a median drop of 15 percentage points (pp) and a mean drop of 21 pp. The penalty persists under a stricter threshold: among the 32 model–game pairs with baseline accuracy of at least 80%, the median CF drop is 20 pp. Only 13 of these 32 pairs maintain CF accuracy at or above 80%, and 25 remain at or above 50%.

Thus, the CF penalty is not merely an artifact of weak baseline performance: even models with strong baseline accuracy often lose accuracy after a single rule change.

5.2 Decomposing the CF Gap

We next inspect CF behavior using observable diagnostics to examine if models might still be operating in baseline instead of the CF world after rules change. We ask whether the model response mentions the altered rule and whether the model's proposed solution is legal under the baseline and CF verifiers. For this, we analyze the visible model response, i.e., the text returned to the user.

Mentioning the altered rule is often associated with, but does not guarantee, correctness. Table 3 splits CF responses according to whether the visible text mentions the altered rule. For DS-R1, Sonnet, Grok, Gemma, Opus, and Qwen3, responses that mention the altered rule are substantially more accurate, with gains of +9 to +42 pp. However, this pattern is not universal: DS-V3, Lla-3.3, and Gemini are less accurate when the rule is mentioned. This suggests that explicitly stating the rule change is helpful for many models, but does not by itself ensure correct execution: even when the altered rule appears in the response, models may still fail to apply it consistently.

Step-by-step verification reveals reversion to the baseline game. For move-sequence games, we simulate the model's proposed moves step by step under both the baseline and CF rules. This identifies failures where the sequence is legal for the baseline game but violates the rules of the CF world. Hanoi-adjacent is a clean diagnostic: direct $A \leftrightarrow C$

[3] Appendix E and F report the full model configuration.
[4] Token-usage results in Appendix D.4.

Game	Variant	Opus	Sonnet	Grok	Gemini	Qwen3	DS-V3	DS-R1	Lla-3.3	GPT120	GPT20	Gemma	OLM32
Sudoku	**baseline**	82	54	0	0	0	0	0	0	51	42	2	0
	CF1 double-8	−39	−28	0	0	0	0	0	0	−51	−42	−2	0
	CF2 shifted-boxes	+7	+19	0	0	0	0	0	0	−51	−42	−2	0
	CF3 11-19	+6	−13	0	0	0	0	0	0	−45	−42	−2	0
	CF4 hex 16×16	−82	−54	0	0	0	0	0	0	−51	−42	−2	0
Cryptarithmetic	**baseline**	100	100	46	76	92	90	78	0	96	95	98	0
	CF1 base-9 add	−4	−1	−16	−18	−3	−14	−2	0	−1	−12	−4	0
	CF2 base-9 sub	0	0	+12	−49	+5	−40	−2	0	−27	−55	−38	0
	CF3 base-9 mixed	0	−1	−36	−69	−10	−45	−25	0	−51	−74	−41	0
Countdown	**baseline**	90	99	7	96	4	97	27	10	94	62	99	42
	CF1 base-9	+6	0	+58	−3	+88	+3	−3	+25	−1	+1	0	−33
	CF2 no-subtraction	+5	−7	−4	−28	+27	−42	−14	−6	−43	−37	−18	−22
	CF3 use-all-numbers	−1	−6	+1	−10	0	−18	+13	−9	−18	−19	−10	−13
	CF4 multi-target	+10	0	+86	+3	+96	+3	−27	+46	+3	+25	+1	−27
Game-24	**baseline**	89	92	30	89	52	92	15	15	80	64	88	33
	CF1 base-9	+8	+6	+4	−11	+21	−27	−7	−6	−30	−28	−16	−21
	CF2 each-op-once	+6	+5	−9	−6	+16	−6	+9	+3	−16	−9	−2	+11
	CF3 prime-target	+6	+6	+4	−1	−42	−6	+1	+2	−1	−16	+3	+26
Water-Jugs	**baseline**	89	94	25	61	82	100	80	25	96	88	100	41
	CF1 gcd-breaker	−45	−36	−23	−16	−23	−59	−30	−14	−65	−32	−63	−24
	CF2 entropy-trap	−30	−39	−25	−8	−26	−57	−25	−7	−59	−47	−52	−31
Hanoi	**baseline**	67	83	20	90	37	83	0	50	60	43	13	23
	CF1 four-pegs	+20	−6	−13	−17	−17	−46	0	−50	+17	−6	−10	−3
	CF2 adjacent	−17	−43	−20	−80	−37	−80	0	−50	−37	−13	−13	−23
Checker-Jumping	**baseline**	72	32	0	0	0	0	0	0	12	8	4	0
	CF1 asymmetric	−35	−19	0	+2	0	0	0	0	−12	−8	−4	0
	CF2 jump-two	0	−12	0	+8	0	0	0	0	−4	−8	0	0
	CF3 two-empty	−60	−32	0	0	0	0	0	0	−12	−8	−4	0
River-Crossing	**baseline**	100	75	0	10	0	20	10	0	5	5	65	0
	CF1 neutral	−5	−15	+5	+20	0	+20	+5	0	+40	+10	−5	0
Overall (all 30 cells)		**80**	**75**	**20**	**49**	**43**	**53**	**36**	**10**	**54**	**42**	**56**	**15**

Table 2: Exact-match accuracy on `CounterPlay`. Baseline rows report raw accuracy (%). Counterfactual (CF) rows report the paired change from the corresponding baseline for the same model and game, $\Delta = \mathrm{Acc}_{\mathrm{CF}} - \mathrm{Acc}_{\mathrm{base}}$, in percentage points. The Overall row reports mean raw accuracy across all 30 game–variant cells. Blue shading marks higher baseline accuracy; green/red shading marks positive/negative CF deltas, with color intensity normalized within each game block. Model abbreviations: Opus = Claude-Opus-4.7, Sonnet = Claude-Sonnet-4.6, DS-V3 = DeepSeek-V3.2, DS-R1 = DeepSeek-R1, Lla-3.3 = Llama-3.3-70B, and OLMo32 = OLMo-3.1-32B. For space, Llama-3.1-8B, Qwen3.5-9B, and OLMo-2-7B are omitted; Appendix Table 7 reports all models.

Model	n_{silent}	*silent* acc	n_{ack}	*ack* acc	Δ
DS-R1	697	5.6%	1093	47.8%	**+42.2**
Sonnet	857	55.8%	933	86.7%	**+30.9**
Grok	1194	10.3%	596	39.9%	**+29.6**
Gemma	619	40.5%	1171	57.8%	+17.3
Opus	1488	75.1%	302	89.7%	+14.6
Qwen3	592	37.5%	1198	46.6%	+9.1
DS-V3	510	50.0%	1280	46.2%	−3.8
Lla-3.3	727	13.2%	1063	6.9%	−6.3
Gemini	1055	49.9%	735	39.9%	−10.0
GPT-OSS-*	≥ 99% silent		(no externalization)		—

Table 3: Visible rule mention and CF accuracy. CF responses are split by whether they mention the altered rule (*ack*, acknowledged) or not (*silent*). n_{silent} and n_{ack} give response counts; accuracies are computed within each bucket. Δ is $\mathrm{Acc}_{\mathrm{ack}} - \mathrm{Acc}_{\mathrm{silent}}$ in % points. GPT-OSS-* models are silent at least 99% of CF responses.

moves are legal in standard Hanoi but forbidden when only adjacent-peg moves are allowed. Table 5 shows large model differences. All incorrect Llama-3.3 responses contain a forbidden direct move, and DS-V3 and Grok also revert frequently (59% and 50%). By contrast, Opus, Sonnet, Gemini, and GPT-OSS-20B are below 14%, suggesting that, by this diagnostic, their failures are less often simple baseline reversion and more often breakdowns under the altered rule.

Exact match hides different strategies. Some correct CF answers use shortcuts rather than solving directly in the altered rule space. In the base-9 variants of Countdown and Game-24, 98–99% of correct answers use a wrapper strategy: convert the base-9 inputs to base 10, solve the familiar problem, and convert the answer back. This is aggregated[5] across all models that produce any correct CF answer on the base-9 variants of Countdown and Game-24. For move-sequence CFs, a small proportion (at most 6.5%) of failures are locally valid but incomplete searches: all moves are legal under the CF rules, but the goal is not reached. We also observe baseline-strategy bias among correct solutions, such as four-peg Hanoi answers that ignore the extra peg and use the standard three-peg

[5]Appendix G reports the per-model breakdown.

recursion (in this CF, the baseline-strategy solutions are still valid in the CF game).

5.3 Which CF Properties Predict Difficulty?

Table 4 characterizes each CF variant and asks which properties of the rule change are associated with performance degradation. Each row is one variant of one game. We annotate the CF along three descriptive axes: **structure**, the type of rule change; **complexity**, how much the change alters the required solving effort; and **proximity**, how far the variant departs from the baseline in surface form, operator semantics, and invariants. The empirical quantity is Δ_{cmp}: for each model achieving at least 50% accuracy on the corresponding baseline game, we compute the paired difference $\text{Acc}_{\text{CF}} - \text{Acc}_{\text{base}}$ and average these differences across models. Thus, negative values indicate a drop in accuracy under the CF rules for models that can solve the baseline game. Given the small class sizes and their partial dependence on specific games, we treat these annotations as a qualitative exploration of recurring patterns.

Structure: We define each structure class (Table 4): **Symbol mapping** (map.) relabels the surface alphabet or positions but leaves the underlying rules and search space intact (e.g., Sudoku 11-19). **Translation wrapper** (wrap.) re-expresses the puzzle in a new representation, typically reducible to the baseline by a decode-solve-encode pipeline (e.g., Cryptarithmetic base-9 addition). **Relaxation** (relax.) adds a resource or removes a constraint, so any baseline solution remains valid (e.g., Hanoi four-pegs). **Tightening** (tight.) removes a resource or forbids a class of moves, shrinking the legal solution set (e.g., Hanoi-adjacent). **Semantic or invariant change** (sem.) redefines the operator semantics or breaks an invariant that the baseline strategy relies on (e.g., Cryptarithmetic base-9 sub/mixed). *Wrapper* and *relaxation* variants are often among the smallest drops, such as Countdown multi-target ($+3.9$ pp), Cryptarithmetic base-9 addition (-7.1 pp), Sudoku shifted-boxes (-8.3 pp), River-Crossing neutral (-8.3 pp), and Checker jump-two ($+0.0$ pp). By contrast, *semantic or invariant-change* variants show larger drops in this set: Checker two-empty (-60.0 pp), Water-Jugs gcd-breaker (-40.9 pp), Water-Jugs entropy-trap, which imposes an evaporation tax (-38.2 pp), Checker asymmetric (-35.3 pp), and Cryptarithmetic mixed base-9 operations (-35.7 pp). However, rule type alone does not determine difficulty. *Tightening* variants, conditioned on baseline competence, range from near-zero loss in Game-24 each-op-once (-1.6 pp) to large drops in Hanoi-adjacent (-51.1 pp). Similar patterns happen with *symbol mapping*.

Complexity: Within the same game, changing solving effort can dominate the effect of broad rule type. In Hanoi, the 4-peg relaxation produces a moderate drop (-13.9 pp), while the adjacent-move variant drops by -51.1 pp as the altered transition graph makes solutions longer. Countdown multi-target shows the opposite pattern: splitting the numbers into two smaller searches yields a positive Δ_{cmp} of $+3.9$ pp. Large drops also occur when a familiar surface hides a changed rule or invariant. Cryptarithmetic base-9 addition has a small drop (-7.1 pp), but mixed variant, which requires coordinating carries and borrows, drops by -35.7 pp. Water-Jugs gcd-breaker changes the reachability invariant and drops by -40.9 pp. Conversely, surface novelty alone is not sufficient: Sudoku 11–19 changes the symbol alphabet and drops by -17.3 pp, whereas Sudoku hex drops by -62.3 pp because it also expands the grid to 16×16.

Proximity. Largest drops reflect two main patterns. Some variants are *search-bound*: they increase effective search burden, as in Sudoku hex and Hanoi-adjacent. This is related to complexity. Others are *constraint- or invariant-bound*: they preserve a familiar task surface but change the constraints or operations that make the baseline strategy invalid, as in Sudoku double-8, Water-Jugs CFs, and Cryptarithmetic mixed base-9 operations. This is what the proximity axis is intended to capture. Conversely, variants whose proximity-distance is dominated by surface form (e.g., Sudoku 11-19) show much smaller drops.

5.4 Does Scaling Up or Adding Explicit Reasoning Close the Gap?

We finally ask whether two model-side factors narrow the CF gap: scale and reasoning. We analyze scale through four open-weight model families with two sizes, and reasoning through models that expose a reasoning configuration or trace. We also test whether longer outputs are linked with higher accuracy. Overall, scale increases baseline competence and reasoning helps on some search-heavy variants, but neither eliminates the CF penalty.

Game	CF	Structure		Complexity	Proximity	Δ cmp
Sudoku	CF1	tight.	digit 9 removed. 8 appears twice in each row/col/box	shallow, standard recipe misleads	surf L, op-sem L, inv M	−39.3
	CF2	map.	3×3 boxes shifted with toroidal wrap-around	shallow, no famous recipe	surf L, op-sem N, inv N	−8.3
	CF3	map.	digit alphabet $\{1,\ldots,9\}$ relabeled to $\{11,\ldots,19\}$	shallow, decode is trivial	surf H, op-sem N, inv N	−17.3
	CF4	map.	hex alphabet $\{0,\ldots,9,$ A–F$\}$ on 16×16 grid	deep, much larger search space	surf H, op-sem N, inv N	−62.3
Crypt.	CF1	wrap.	$W_1+W_2=R$ in base 9. Carries shift from 10 to 9	medium, OOD base but reducible	surf L, op-sem M, inv L	−7.1
	CF2	sem.	subtraction $W_1-W_2=R$ in base 9. Borrow shifts	medium, OOD borrow semantics	surf L, op-sem H, inv M	−23.4
	CF3	sem.	chained $W_1+W_2-W_3=R$ with carry and borrow mixed	med-deep, chained column ops	surf L, op-sem H, inv H	−35.7
Countd.	CF1	wrap.	all source numerals and the target written in base 9	shallow-med, decode then solve	surf M, op-sem L, inv L	−3.4
	CF2	tight.	subtraction operator forbidden. Only $+,\times,\div$ allowed	shallow, no standard recipe	surf L, op-sem M, inv M	−25.4
	CF3	tight.	every source number used exactly once in expression	medium, tighter search constraint	surf L, op-sem N, inv L	−11.9
	CF4	relax.	two fixed groups of three. Each group reaches own target	shallow, two smaller searches	surf L, op-sem N, inv N	+3.9
Game-24	CF1	wrap.	four operands and target in base 9 ("24"→"26")	shallow, decode then encode	surf M, op-sem L, inv L	−13.4
	CF2	tight.	no operator repeated. At most one of $+,-,\times,\div$	shallow, tighter operator set	surf L, op-sem N, inv L	−1.6
	CF3	sem.	target is a prime $\neq 24$. Multiset makes 24 unreachable	shallow, goal value moved	surf L, op-sem N, inv M	−5.8
W-Jugs	CF1	sem.	target $\notin \gcd(A,B)\mathbb{Z}$. Reachable only via evaporation tax	medium, classic invariant broken	surf L, op-sem L, inv H	−40.9
	CF2	sem.	pours cost a 1-unit evaporation tax. Optimal opening differs	medium, baseline first move misleads	surf L, op-sem M, inv M	−38.2
Hanoi	CF1	relax.	fourth peg available (Frame-Stewart beats 2^n-1)	medium, well-known recipe	surf L, op-sem L, inv L	−13.9
	CF2	tight.	pegs in a line. Only adjacent-peg moves allowed	deep (3^n-1), no famous recipe	surf L, op-sem M, inv M	−51.1
Checker	CF1	sem.	N red vs. $N+k$ blue frogs. Left–right symmetry broken	medium, no standard recipe	surf L, op-sem L, inv M	−35.3
	CF2	relax.	adds jump over two consecutive opposite frogs	medium, move graph differs	surf L, op-sem M, inv L	+0.0
	CF3	sem.	two empty cells between color groups instead of one	medium, no standard recipe	surf L, op-sem L, inv M	−60.0
River	CF1	relax.	extra neutral traveller with no predator/prey constraints	shallow, baseline policy still valid	surf L, op-sem N, inv L	−8.3

Table 4: CF properties and baseline-conditioned difficulty. Each row is one CF variant. **Structure** gives the rule-change type: map. symbol mapping, wrap. translation wrapper, relax. relaxation, tight. tightening, and sem. semantic or invariant change. **Complexity** describes how much the CF rule changes the required solving effort, from shallow cases to deeper search or longer plans. **Proximity** rates how far the CF departs from the original game along three dimensions: surface form (*surf*), operator semantics (*op-sem*), and invariants (*inv*); N/L/M/H denote none/low/medium/high distance. $\Delta_{\text{cmp}} = \text{Acc}_{\text{CF}} - \text{Acc}_{\text{base}}$ (cmp = competent) is the paired accuracy change in percentage points, averaged over models with baseline accuracy at least 50% on the corresponding game. Negative values indicate CF accuracy drops; redder cells indicate larger drops. Abbreviations: Crypt. = Cryptarithmetic, Countd. = Countdown, W-Jugs = Water-Jugs, Checker = Checker-Jumping, River = River-Crossing.

Scale improves coverage, but does not guarantee CF robustness. Four open-weight families in our suite include two model scales. In all four cases, the larger model improves overall accuracy (Table 7 in the Appendix): OLMo-2-7B to OLMo-3.1-32B, Llama-3.1-8B to Llama-3.3-70B, Qwen3.5-9B to Qwen3-235B, and GPT-OSS-20B to GPT-OSS-120B. The gains are especially large for Qwen: the 235B model reaches non-trivial benchmark performance, whereas the 9B model remains near zero.

However, these gains mostly expand the number of games on which a model has usable baseline competence; they do not remove the CF penalty. Under the same baseline-conditioned gap calculation used above, the GPT-OSS family shows nearly unchanged drops across scale: roughly -29 pp for GPT-OSS-20B and -30 pp for GPT-OSS-120B. For Llama and OLMo, both sizes remain below the 50% baseline-competence threshold on most games, leaving few settings where a meaningful paired CF gap can be measured. Qwen3-235B is the clearest case where scaling creates usable baseline competence, but its CF performance still tracks its baseline performance rather than showing immunity to altered rules. Thus, scale helps models cover more of the benchmark, but does not by itself make them robust to CF rule changes.

Model	n	forbidden %	Model	n	forbidden %
Lla-3.3	30	**100%**	GPT-OSS-120B	23	22%
DS-V3	29	59%	Qwen3	30	20%
Grok	30	50%	Opus	15	13%
DS-R1	30	30%	Sonnet	18	11%
Gemma	30	30%	GPT-OSS-20B	21	10%
			Gemini	27	4%

Table 5: Default-rule reversion on Hanoi-adjacent. For incorrect CF responses, n is the response count; **forbidden %** is the share with a direct $A \leftrightarrow C$ disk move, which is legal in standard Hanoi but illegal in the CF variant, where only adjacent-peg moves are allowed.

Reasoning-capable models help on some search-heavy variants, but are not sufficient. Seven evaluated models expose a reasoning configuration or trace: Claude-Opus, Claude-Sonnet, GPT-OSS-120B, GPT-OSS-20B, DeepSeek-V3.2, DeepSeek-R1, and Gemini-2.5-Flash. These models include the strongest performers and account for all non-zero results on Hanoi-adjacent, the search-heavy variant in which only adjacent-peg moves are allowed. Opus and Sonnet lead on this variant, followed by GPT-OSS, Gemini, and a small non-zero result for DeepSeek-V3.2; DeepSeek-R1 and the non-reasoning models remain at zero.

At the same time, reasoning is neither sufficient for strong performance nor necessary for all parts of the benchmark. Gemma-3-27B does not expose such a reasoning configuration or trace, yet reaches 56% overall accuracy and outperforms several reasoning-capable models in aggregate. Conversely, DeepSeek-R1 produces one of the longest traces in the suite, but underperforms DeepSeek-V3.2 on both baseline and CF variants. This suggests that reasoning capabilities help on some search-heavy variants, but do not automatically solve invariant- or constraint-bound failures, where the main challenge is to apply the altered rule consistently rather than only to search longer.

5.5 Effort vs. Correctness on Hanoi-adjacent

A recurring question with reasoning-capable models is whether spending more tokens on a hard problem actually helps. We test this on Hanoi-adjacent, a CF in which only adjacent-peg moves are legal and the optimal plan grows from 2^n-1 to 3^n-1. Each model is run 5 independent times on each puzzle (§4), providing correct and incorrect attempts that can be compared in token cost. If extra effort helps, we expect correct attempts to use *more* tokens than incorrect ones. If effort and outcome are unrelated, we expect no consistent gap.

Model	$\bar{t}_{\text{ok}}$	$\bar{t}_{\text{err}}$
Claude-Opus-4.7	9,762	6,269
Claude-Sonnet-4.6	14,291	13,764
GPT-OSS-120B	668	1,731
GPT-OSS-20B	982	745
Gemini-2.5-Flash	152	328
DeepSeek-V3.2	1,874	1,279
DeepSeek-R1	—	18,091
Gemma-3-27B	—	4,192
Grok-4.1-fast	—	1,114
Llama-3.3-70B	—	907
Qwen3-235B	—	5,825

Table 6: Mean output tokens on correct ($\bar{t}_{\text{ok}}$) vs. incorrect ($\bar{t}_{\text{err}}$) runs of Hanoi-adjacent. "—" marks models with no correct trial. Tokens include hidden reasoning where billed (Opus, Sonnet, DeepSeek-R1, GPT-OSS); otherwise visible characters / 4.

Effort does not predict success. Table 6 reports each model's mean output tokens on correct vs. incorrect attempts. For most models, the picture is the opposite of what an effort hypothesis would predict: the failed attempts use as many or more tokens than the successful ones. The clearest case is Claude-Sonnet on the $n = 6$ puzzles, where the three correct runs averaged 28.5k tokens against 32.8k for the two failures. Five models (DeepSeek-R1, Gemma, Grok, Llama-3.3-70B, Qwen3-235B) score zero on every puzzle despite 1k-18k tokens per attempt, with DeepSeek-R1 the most extreme. Where token spend is higher on correct runs (e.g. Claude-Opus), inspection of the puzzles shows the gap reflects puzzle hardness rather than effort: Opus succeeds on smaller-n instances whose plans still take more tokens than its failed attempts on n=7. In short, longer outputs co-occur with harder puzzles, not with better outcomes at a given puzzle.

6 Conclusion

We introduced `CounterPlay`, a benchmark of verifiable counterfactual games for evaluating if LLMs can adapt to explicitly altered rules in familiar task environments. Across 8 games, 22 counterfactual variants, and 2,365 puzzles with deterministic verifiers, we find that counterfactual rule changes remain challenging even for models that solve the corresponding baseline games. Our analyses show that models may revert to baseline rules, mention the altered rule but fail to execute it, or rely on wrapper strategies that reduce the variant to a familiar baseline task. While scale and explicit reasoning improve baseline competence and some search-heavy variants, they do not reliably close the counterfactual gap. `CounterPlay` offers a controlled testbed for studying counterfactual reasoning in LLMs.

Limitations

Visible-response diagnostic for rule mention. For the analysis of whether models mention altered rules, the reasoning channel is not available across all APIs in our suite, so we restrict the diagnostic to the visible response for uniformity. This is an imperfect signal: a model could internally use the new rule without explicitly restating it, so a silent response does not by itself imply non-adaptation. We treat the rule mentioned as a necessary but not sufficient correlate of rule-following.

Single-agent games. The set of games in `CounterPlay` is limited to single-agent ones. This is intentional, as multiple-agent games make the comparison of models and variants difficult because they require agents other than the LLMs themselves, playing against them.

Verifier-restricted notion of correctness. "Correct" in `CounterPlay` means satisfying a deterministic game-specific verifier under the relevant rules. This deliberately excludes notions of solution quality, elegance, or strategic insight (e.g., whether a four-peg Hanoi solution exploits the Frame-Stewart shortcut, or solves the variant by ignoring the extra peg). Section 5.2 flags the latter as baseline-strategy bias, but our headline accuracy numbers do not penalize it. Benchmarks that aim to measure whether models exploit the altered rule structure (rather than merely satisfy the verifier) would require richer scoring than an exact match.

References

Ishwar Balappanawar, Vamshi Bonagiri, Anish Joishy, Manas Gaur, Krishnaprasad Thirunarayan, and Ponnurangam Kumaraguru. 2025. If pigs could fly... can llms logically reason through counterfactuals?

Ruth M. J. Byrne. 2016. Counterfactual thought. *Annual review of psychology*, 67:135–57.

Guanyu Chen, Peiyang Wang, Yizhou Jiang, Yuqian Liu, Chujie Zhao, Ying Fang, Tianren Zhang, and Feng Chen. 2025. Exploring the hidden reasoning process of large language models by misleading them. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 20263–20278, Suzhou, China. Association for Computational Linguistics.

Yuefei Chen, Vivek Singh, Jing Ma, and Ruixiang Tang. 2026. Counterbench: Evaluating and improving counterfactual reasoning in large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40:30350–30358.

François Chollet. 2019. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*.

Dimitrios Georgousis, Maria Lymperaiou, Angeliki Dimitriou, Giorgos Filandrianos, and Giorgos Stamou. 2026. Evaluating counterfactual strategic reasoning in large language models. *arXiv preprint arXiv:2603.19167*.

Pavel Golikov, Evgenii Opryshko, Gennady Pekhimenko, and Mark C Jeffrey. 2026. Robust reasoning benchmark. *arXiv preprint arXiv:2604.08571*.

Jiayi Gui, Yiming Liu, Jiale Cheng, Xiaotao Gu, Xiao Liu, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2025. LogicGame: Benchmarking rule-based reasoning abilities of large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 1474–1491, Vienna, Austria. Association for Computational Linguistics.

Nicole Van Hoeck, Patrick D. Watson, and Aron Keith Barbey. 2015. Cognitive neuroscience of human counterfactual reasoning. *Frontiers in Human Neuroscience*, 9.

Kaixuan Huang, Jiacheng Guo, Zihao Li, Xiang Ji, Jiawei Ge, Wenzhe Li, Yingqing Guo, Tianle Cai, Hui Yuan, Runzhe Wang, Yue Wu, Ming Yin, Shange Tang, Yangsibo Huang, Chi Jin, Xinyun Chen, Chiyuan Zhang, and Mengdi Wang. 2025. Mathperturb: benchmarking llms' math reasoning abilities against hard perturbations. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org.

Sai Adith Senthil Kumar, Hao Yan, Saipavan Perepa, Murong Yue, and Ziyu Yao. 2025. Can LLMs simulate personas with reversed performance? a systematic investigation for counterfactual instruction following in math reasoning context. In *First Workshop on Social Simulation with LLMs*.

Eldar Kurtic, Amir Moeini, and Dan Alistarh. 2024. Mathador-LM: A dynamic benchmark for mathematical reasoning on large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17020–17027, Miami, Florida, USA. Association for Computational Linguistics.

Chenxi Li, Yuanhe Tian, Zhaxi Zerong, Yan Song, and Fei Xia. 2024a. Challenging large language models with new tasks: A study on their adaptability and robustness. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8140–8162, Bangkok, Thailand. Association for Computational Linguistics.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024b. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2961–2984, Bangkok, Thailand. Association for Computational Linguistics.

Jiayu Liu, Zhenya Huang, Wei Dai, Cheng Cheng, Jinze Wu, Jing Sha, Song Li, Qi Liu, Shijin Wang, and Enhong Chen. 2025. Cogmath: Assessing LLMs' authentic mathematical ability from a human cognitive perspective. In *Forty-second International Conference on Machine Learning*.

Keith Markman, Matthew Lindberg, and Adam Galinsky. 2007. Implications of counterfactual structure for creative generation and analytical problem solving. *Personality & social psychology bulletin*, 33:312–24.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2025. GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*.

Dayyán O'Brien, Barry Haddow, Emily Allaway, and Pinzhen Chen. 2025. Mathemagic: Generating dynamic mathematics benchmarks robust to memorization. *arXiv preprint arXiv:2510.05962*.

Judea Pearl and Dana Mackenzie. 2018. *The Book of Why: The New Science of Cause and Effect*. Basic Books.

Jeffrey Seely, Yuki Imajuku, Tianyu Zhao, Edoardo Cetin, and Llion Jones. 2026. Sudoku-bench: Evaluating creative reasoning with sudoku variants.

Parshin Shojaee, Seyed Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2026. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, L. Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *ArXiv*, abs/1712.01815.

Shashank Sonkar, Naiming Liu, MyCo Le, and Richard Baraniuk. 2024. MalAlgoQA: Pedagogical evaluation of counterfactual reasoning in large language models and implications for AI in education. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15554–15567, Miami, Florida, USA. Association for Computational Linguistics.

Fien van Wetten, Aske Plaat, and Max van Duijn. 2026. Baba is llm: Reasoning in a game with dynamic rules. In *Computational Intelligence*, pages 60–77, Cham. Springer Nature Switzerland.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2024. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1819–1862, Mexico City, Mexico. Association for Computational Linguistics.

Kai Yan, Yufei Xu, Zhengyin Du, Xuesong Yao, Zheyu Wang, Xiaowen Guo, and Jiecao Chen. 2025. Recitation over reasoning: How cutting-edge language models can fail on elementary school-level reasoning problems? In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 272–291, Mumbai, India. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics.

Tong Yu, Yongcheng Jing, Xikun Zhang, Wentao Jiang, Wenjie Wu, Yingjie Wang, Wenbin Hu, Bo Du, and Dacheng Tao. 2025. Benchmarking reasoning robustness in large language models. *arXiv preprint arXiv:2503.04550*.

Ruiwen Zhou, Wenyue Hua, Liangming Pan, Sitao Cheng, Xiaobao Wu, En Yu, and William Yang Wang. 2025. RuleArena: A benchmark for rule-guided reasoning with LLMs in real-world scenarios. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 550–572, Vienna, Austria. Association for Computational Linguistics.

A The games

Here, each game is introduced in more detail. The counterfactual variants are then described in enough detail to make the rule change concrete.

Math-driven games

Sudoku. The canonical Sudoku puzzle is a 9×9 grid divided into nine 3×3 boxes. Some cells are pre-filled (*givens*) while the solver must fill in every empty cell with a digit from 1 to 9 so that each row, each column, and each 3×3 box contains the digits 1-9 exactly once. The puzzle has a unique solution. *CF1 (double-8):* the symbol 9 is removed, and instead the digit 8 appears *twice* in every row, column, and box. The grid is still 9×9 but the alphabet is now $\{1, \ldots, 8\}$. It means the alphabet is the multiset $\{1, 2, 3, 4, 5, 6, 7, 8, 8\}$. The digit 8 has multiplicity 2. *CF2 (shifted boxes):* the rows and columns rules are unchanged, but the nine 3×3 boxes are shifted by one cell with toroidal wrap-around, each box now wraps over the right/bottom edge. The standard Sudoku boxes are anchored at the top left of each 3×3 block. In CF2 every box shifts one cell to the right (and

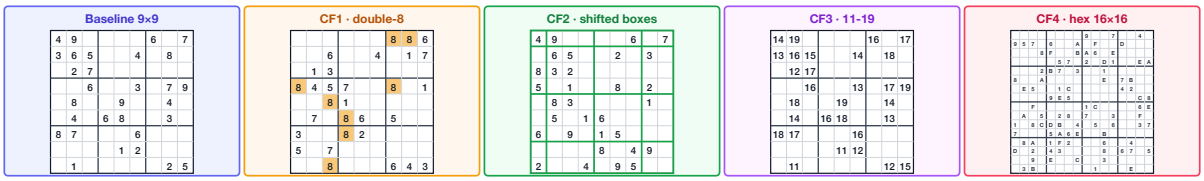

Figure 1: Sudoku: one real puzzle per variant with its verified solution.

Figure 2: Cryptarithmetic: one real equation per variant with the verifying digit assignment.

one cell down), with positions wrapping toroidally over the right and bottom edges. The top left box, for example, now covers rows 1 to 3 and columns 1 to 3 instead of rows 0 to 2 and columns 0 to 2. The rightmost box of the top band wraps so that its right column lies at column 0. Row and column rules are unchanged. *CF3 (11-19):* the symbol set $\{1, \ldots, 9\}$ is relabeled to $\{11, 12, \ldots, 19\}$. *CF4 (hex* 16×16*):* the grid grows to 16×16 with 4×4 boxes and the symbol set $\{0, 1, \ldots, 9, \mathrm{A}, \ldots, \mathrm{F}\}$. This is a much larger search problem and serves as a stress test on grid handling.

Cryptarithmetic. A cryptarithm is an equation such as $\mathtt{SEND} + \mathtt{MORE} = \mathtt{MONEY}$ in which each letter stands for a distinct digit. The task is to recover the letter-to-digit assignment that makes the equation arithmetically true. The baseline uses base-10 addition with the format $W_1 + W_2 = R$. *CF1 (base-9 add):* the same $W_1 + W_2 = R$ format but every column sum carries at 9 instead of 10, and the symbol 9 never appears. *CF2 (base-9 sub):* the equation form becomes $W_1 - W_2 = R$, with borrowing of 9 (not 10) in base-9 subtraction. *CF3 (base-9 mixed):* the form is $W_1 + W_2 - W_3 = R$, requiring both carries and borrows to be tracked in a single chained computation in base 9.

Countdown. Countdown is a British TV puzzle: the solver is given a target number (typically 100-999) and six source numbers, and must reach the target using addition, subtraction, multiplication, and division on a subset of the sources. intermediate results must remain non-negative integers. Our baseline follows these rules. *CF1 (base-9):* every numeral in the prompt and in the answer expression, operands, target, and intermediate literal, is written in base 9. *CF2 (no subtraction):* the $-$ operator is forbidden. Only $+, \times, \div$ are allowed, and instances are filtered so the standard rule's subtraction-using solution genuinely fails. *CF3 (use-all-numbers):* the solver must include all six source numbers in the expression exactly once (instead of any non-empty subset). *CF4 (multi-target):* the six source numbers are split into two fixed groups of three, each of which must reach its own target.

Game-24. Given four numbers , the solver must combine them with $+, -, \times, \div$ and parentheses to reach the target value 24. Each number is used once. *CF1 (base-9):* all four operands and the target "24" are presented as base-9 strings ("24" decimal is "26" in base 9), and the answer expression must use base-9 literals. *CF2 (each operator once):* no operator may be repeated. *CF3 (prime target):* the target is a prime number chosen so the multiset of source values does *not* also reach 24 under standard rules, ensuring the prime target is the only correct goal.

Water-Jugs. Two jugs of fixed integer capacities start empty. The moves available are fill-A, fill-B, empty-A, empty-B, pour-A-to-B, and pour-B-to-A. The goal is to reach a specified target volume in either jug. Our baseline uses the standard (lossless) rules: a pour transfers water exactly, with no loss, until either the source is empty or the destination is full. Under these rules, the reachable volumes are exactly the non-negative multiples of $\gcd(A, B)$ up to $\max(A, B)$.

Both counterfactual variants share an added rule that we call the *evaporation tax*: every pour move loses 1 unit of water (the source loses $\min(\text{Source}, \text{TargetSpace} + 1)$ units, of which 1 unit evaporates and the rest reaches the destination).

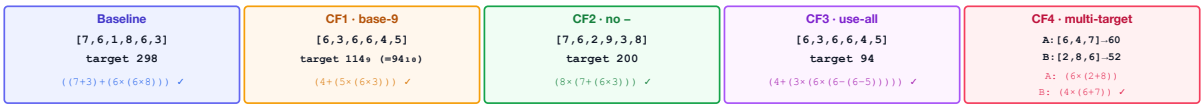

Figure 3: Countdown: one real instance per variant with a witnessing expression.

Figure 4: Game-24: one real instance per variant with a witnessing expression.

We refer to the taxed regime as "Entropy Jugs." *CF1 (GCD-breaker):* puzzles are constructed so $\gcd(A, B) > 1$ and the target is not a multiple of $\gcd(A, B)$, making the target literally unreachable under the classic baseline rules. The evaporation tax breaks the GCD constraint and is what enables a solution to exist at all. *CF2 (entropy trap):* both the classic and the taxed rules can reach the target, but the optimal first move differs between them. A solver relying on classic intuition picks the wrong opening pour.

Validity (each puzzle has at least one solution) is enforced per game by running the verifier during generation and dropping unsolvable instances. Uniqueness of the *solution* is required only for Sudoku and Cryptarithmetic, where the generator samples candidates and keeps only those with exactly one solution (checked by a back tracking counter). The other games enforce uniqueness of the *goal* (one target volume for Water-Jugs, one final stack arrangement for Tower-of-Hanoi, etc.) but accept any plan or expression that reaches that goal under the variant's rules, and the verifier replays the model's output to score correctness.

Non-math games

The three non-math games, Tower-of-Hanoi, Checker-Jumping, and River-Crossing, are the same planning puzzles used as scaling probes by Shojaee et al. (2026) (which varied puzzle size rather than rules). We use them here with rule-altered counterfactual variants rather than size-scaled instances.

Tower-of-Hanoi. Classical Hanoi has three pegs and n disks stacked in decreasing order on peg A. The solver must produce a sequence of one-disk moves that ends with all n disks stacked on peg C, never placing a larger disk on a smaller one. *CF1 (four pegs):* a fourth peg D is available. The disk-ordering constraint is unchanged but the minimum-move solution is given by the Frame-Stewart procedure rather than the $2^n - 1$ formula for three pegs. *CF2 (adjacent-only):* the three pegs are arranged in a line and only moves between physically adjacent pegs ($0 \leftrightarrow 1$ and $1 \leftrightarrow 2$) are legal. the direct $0 \leftrightarrow 2$ move is forbidden. The minimum number of moves grows from $2^n - 1$ to $3^n - 1$.

Checker-Jumping. A row of cells holds N red frogs on the left, one empty cell, and N blue frogs on the right. Each frog may slide one cell forward into an empty cell, or jump over exactly one opposite-color frog into an empty cell beyond it. The goal is to swap the two groups. *CF1 (asymmetric counts):* N red frogs face $N + k$ blue frogs (with $k \in \{1, 3, 5\}$), leaving the standard jump rule intact but breaking the left-right symmetry. *CF2 (jump-two):* in addition to the normal one-frog jump, a frog may jump over *two* consecutive opposite-color frogs, landing three cells away. *CF3 (two empty):* the board has two empty cells (between the two color groups) instead of one. all other rules unchanged, but the legal-move graph changes non-trivially.

River-Crossing. In the classical puzzle a farmer must transport a wolf, a goat and a cabbage across a river using a small boat that can hold the farmer plus one item. If left alone together, the wolf eats the goat or the goat eats the cabbage. The task is to produce the sequence of boat trips that gets all four to the other side without any predation. *CF1 (neutral person):* one additional "neutral" person who can travel in any group is added, with no predation involvement. boat capacity is unchanged. The predator/prey constraints among the original characters are unaffected, but the search graph grows because the neutral person opens additional legal

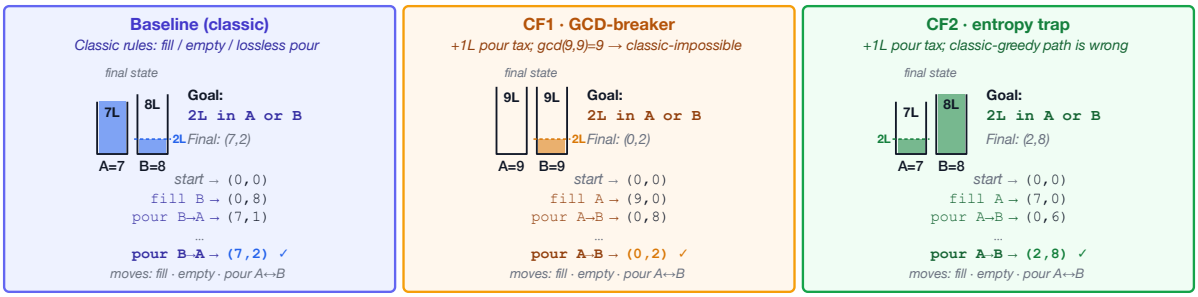

Figure 5: Water-Jugs: one real instance per variant. The first three and the last state of the witnessing path are shown. The baseline uses classic (lossless) rules. Both CF variants add a 1-unit evaporation tax on pours.

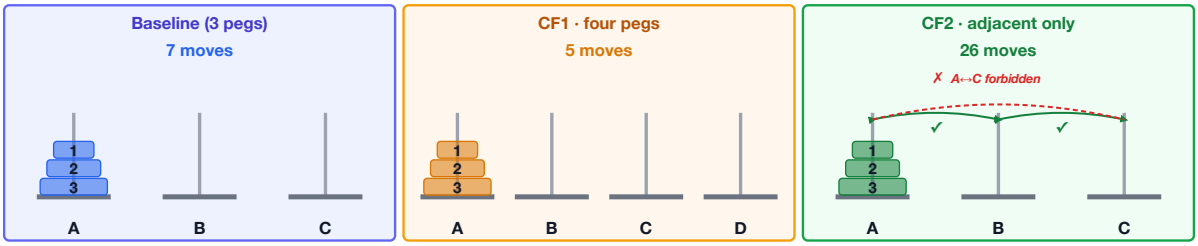

Figure 6: Tower-of-Hanoi: one real instance per variant with the minimum-move count.

configurations.

B Scoring details

This appendix expands the parsing and partial-credit details summarized in Section 3.2.

Parsing. Model outputs are parsed with deterministic game-specific parsers. When possible, prompts request a structured answer format. Otherwise, we extract the answer using a combination of JSON extraction and regular-expression fallbacks. Outputs that cannot be parsed into the required game-specific format are marked as parse failures and counted as incorrect.

Partial credit. In addition to exact correctness, we record partial-credit signals for tasks where exact match can hide meaningful progress. For move-sequence games, a response is marked as *valid* if every move is legal under the relevant rules, even if the sequence does not reach the goal state. Valid-but-incomplete trajectories therefore indicate that the model follows the local game dynamics but fails to complete the search.

For Sudoku, we additionally compute cell-level metrics: the fraction of cells in the model's grid that match the reference solution, and a *respects-givens* indicator measuring whether the model preserves the pre-filled cells. Cell-level correctness is used for the baseline and CF1–CF3, where reference solutions are available. For the 16×16 hexadecimal variant, we report the cell-level score as a structural similarity measure rather than as a complete solvability metric.

C CF axis labels: rationale per variant

This appendix expands Table 4. For each of the 22 CFs we briefly justify the three axis labels: **structure** (map. symbol mapping, wrap. translation wrapper, relax. relaxation, tight. tightening, sem. semantic / invariant break), **complexity** (depth of the search the CF induces, on a shallow→deep scale), and **proximity** (surface, operator-semantics, and invariant distance to the baseline, each rated N/L/M/H).

Sudoku. *CF1 double-8* is tight. because the alphabet shrinks (9 → 8) but each row/col/box must now contain the digit 8 twice. The rule space is strictly more constrained than the baseline. Complexity is shallow but the standard "each digit once" recipe is actively misleading. Surface and operator-semantics distance are low (the grid still looks like Sudoku). Invariant distance is medium because the each-digit-once Latin-square property the baseline strategy relies on is explicitly broken. *CF2 shifted boxes* is map. because only the box geometry is remapped. Rows, columns, and the alpha-

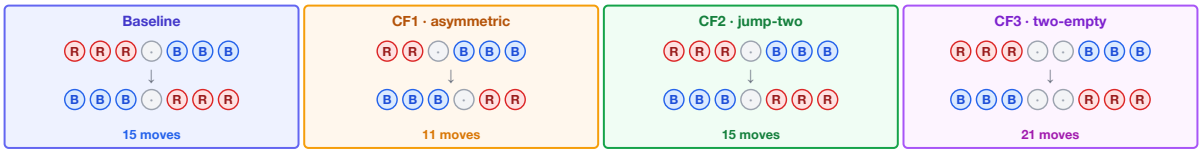

Figure 7: Checker-Jumping: initial and goal configurations per variant.

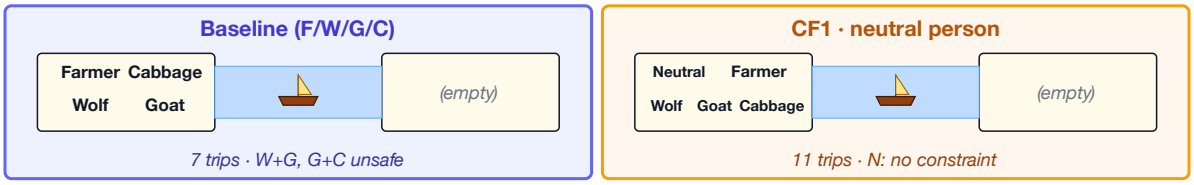

Figure 8: River-Crossing: the start-bank configuration and trip count per variant.

bet are untouched. The search is no harder, but no famous recipe exists for toroidal boxes (no op-sem or invariant change). *CF3 11-19* is map. because every symbol is relabeled from $\{1, \ldots, 9\}$ to $\{11, \ldots, 19\}$ without changing any rule. Surface distance is high (every cell looks different) but operators and invariants are untouched, so decoding back to $\{1, \ldots, 9\}$ is trivial. *CF4 hex* 16×16 is map. for the same reason (alphabet relabeling), but the grid grows to 16×16, which is the only Sudoku CF that materially enlarges the search.

Cryptarithmetic. *CF1 base-9 add* is wrap. because the puzzle shape ($W_1+W_2=R$) is intact. Only the carry threshold changes from 10 to 9, which can be handled by decoding to decimal, solving, and re-encoding. Operator-semantics distance is medium (carries shift) but the invariant "columns sum mod base" still holds, so the wrapper is reversible. *CF2 base-9 sub* is sem. because subtraction in base 9 changes borrow semantics column by column. The simple decimal wrapper no longer applies, and the invariant the model relied on (base-10 borrow) is broken. *CF3 base-9 mixed* is sem. and the hardest of the three: chained $W_1+W_2-W_3=R$ forces simultaneous carry and borrow per column, raising op-sem distance to H and invariant distance to H.

Countdown. *CF1 base-9* is wrap. for the same reason as Crypt CF1: the puzzle structure is unchanged, only the numeral encoding shifts, and decoding is reversible. *CF2 no-subtraction* is tight. because the operator set is strictly reduced ($\{+, -, \times, \div\} \rightarrow \{+, \times, \div\}$). No recipe in pre-training trains the model to plan without subtraction, so the invariant ("always reachable with all four ops") is broken. Op-sem distance is medium because removing an operator from the available set is a structural change to what moves the puzzle exposes, comparable to a new move type appearing in Hanoi-adjacent or Checker jump-two. *CF3 use-all* is tight. because every source number must appear exactly once, removing the freedom to pick a subset. The search becomes a constrained partition rather than a free combination. *CF4 multi-target* is relax. because the six numbers are split into two fixed groups of three, each with its own (smaller) target. Each sub-search is shallower than the baseline.

Game-24. *CF1 base-9* is wrap. for the same wrapper reason. The target "24" decimal becomes "26" in base 9, but the multiset arithmetic is unchanged once decoded. *CF2 each-op-once* is tight. because the operator multiset is restricted (at most one of each), shrinking the expression space without changing what operators mean. *CF3 prime target* is sem. because no move or operator is forbidden; instead the goal value is redefined to a prime $\neq 24$, and the invariant the baseline strategy aims at ("reach 24") no longer applies. This parallels Water-Jugs CF1, where the target is moved off the baseline's reachable set, but the change is softer here (a different reachable value is the goal, not an unreachable one), so invariant distance is medium rather than high.

Water-Jugs. Both CFs add the same 1-unit *evaporation tax* on pours, so both are sem. : pour se-

mantics are redefined and the classic lossless invariant is broken. *CF1 GCD-breaker* pushes the target outside $\gcd(A, B)\mathbb{Z}$, which under the baseline rules is literally unreachable. The tax is what *enables* a solution at all, so invariant distance is H. *CF2 entropy trap* keeps the target inside the reachable set under both rules but is constructed so the move that is optimal under classic rules is no longer optimal under the tax. Op-sem and invariant distances are both M, and the cell is hard because the baseline first-move heuristic actively misleads.

Tower of Hanoi. *CF1 four pegs* is relax. because adding a fourth peg only enlarges the legal move set (Frame–Stewart shortcut beats 2^n-1). The disk-ordering invariant is preserved and the recipe is well known. *CF2 adjacent* is tight. because pegs are laid in a line and only moves between physically adjacent pegs are legal. This preserves disk ordering but redefines what a single "move" can do, pushing the optimal plan from 2^n-1 to 3^n-1. Surface distance is low (one extra sentence) but the depth growth makes it the deepest planning cell in the benchmark. Invariant distance is medium because the depth/recurrence invariant the textbook recursion is built on (2^n-1 moves with a free $A \to C$) is broken.

Checker-Jumping. *CF1 asymmetric* is sem. because no move is removed or forbidden; instead the color-balance invariant (N red vs. N blue) that the standard mirror-symmetric recipe relies on is broken by N vs. $N+k$ frogs. Invariant distance is M. *CF2 jump-two* is relax. because it *adds* a new legal move (jumping over two consecutive opposite frogs) alongside the standard jump, so every baseline solution remains valid. Op-sem distance is M (a new move type), but the puzzle stays solvable with a similar plan, so the gap is near zero. *CF3 two-empty* is sem. because it doesn't remove or forbid moves; it changes the starting configuration so that the one-empty-cell invariant the standard one-gap recipe exploits no longer holds. Surface looks almost identical, but the move graph and reachable-state structure differ, and invariant distance is M.

River-Crossing. *CF1 neutral* is relax. because an extra traveler is added with no predator/prey constraints. The constraint set is strictly weaker and the baseline policy remains a valid solution, so this is the gentlest CF in the table.

D Additional analysis

D.1 Per-variant detail: easy and hard CFs hide inside the average

Averaging accuracy over the CF variants of a game obscures the fact that the variants within a single game span a wide difficulty range. Table 7 gives the exact-match accuracy for every (model, game, variant) cell. Reading down a game block reveals four recurring patterns.

High intra-game variance across counterfactuals For Hanoi, the four-pegs CF reaches as high as 87% (Claude-Opus-4.7) while the adjacent-only CF tops out at 50% (Opus): a roughly 18-point cross-model mean spread on the same game. For Countdown, the multi-target CF is the easiest of the four CFs (mean 63% across the fifteen models, 100% for the top four including Opus) while no-subtraction is the hardest (mean 28%, max 95% for Opus): a 35-point spread. For Cryptarithmetic the base-9 add CF stays within $\Delta \leq 18$pp of its baseline (Gemini-2.5-Flash 76% $\to$ 58%, GPT-OSS-120B 96% $\to$ 95%), but base-9 mixed drops by 36–74pp for the reasoning-capable models (GPT-OSS-20B -74, Gemini -69, GPT-OSS-120B -51). Averaging these together yields a deceptively narrow "CF accuracy" number while the per-variant view recovers the real difficulty curve.

Reading the table per model. Gemma-3-27B is the strongest on math-driven CFs (mean 57.1% across the 16 math CF combinations, edging GPT-OSS-120B's 49.6%) and is the strongest single model on countdown no-subtraction (81%) and game-24 prime-target (91%). GPT-OSS-120B leads on Sudoku partial credit on every 9×9 variant except 11-19, with 43-57% cell-level accuracy on the 9×9 grids (see Table 10). DeepSeek-V3.2 reaches 100% on countdown base-9 and multi-target and has its largest single drop on water-jugs (100 $\to$ 42%, second only to GPT-OSS-120B's 96 $\to$ 34 on the same game). Qwen3-235B's cryptarithmetic base-9 sub (97%) still *exceeds* its own base-10 baseline (92%), the cleanest within-model CF bonus in the table. The sub-10B anchors are essentially at the floor: Llama-3.1-8B's highest entries are countdown multi-target (9%) and hanoi baseline (3%). Qwen3.5-9B hits 17% on cryptarithmetic baseline (but 0% on every cryptarithmetic

Game	Variant	Opus	Sonnet	Grok	Gemini	Qwen3	DSeek-V3	DSeek-R1	Lla-3.3	GPT120	GPT20	Gemma	OLMo32	Lla-3.1	Qwen3.5-9B	OLMo
Sudoku	**baseline**	82	54	0	0	0	0	0	0	51	42	2	0	0	0	0
	CF1	43	26	0	0	0	0	0	0	0	0	0	0	0	0	0
	CF2	89	73	0	0	0	0	0	0	0	0	0	0	0	0	0
	CF3	88	41	0	0	0	0	0	0	6	0	0	0	0	0	0
	CF4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cryptarithmetic	**baseline**	100	100	46	76	92	90	78	0	96	95	98	0	0	17	0
	CF1	96	99	30	58	89	76	76	0	95	83	94	0	0	0	0
	CF2	100	100	58	27	97	50	76	0	69	40	60	0	0	5	0
	CF3	100	99	10	7	82	45	53	0	45	21	57	0	0	0	0
Countdown	**baseline**	90	99	7	96	4	97	27	10	94	62	99	42	1	0	0
	CF1	96	99	65	93	92	100	24	35	93	63	99	9	3	0	1
	CF2	95	92	3	68	31	55	13	4	51	25	81	20	0	0	0
	CF3	89	93	8	86	4	79	40	1	76	43	89	29	1	0	0
	CF4	100	99	93	99	100	100	0	56	97	87	100	15	9	0	0
Game-24	**baseline**	89	92	30	89	52	92	15	15	80	64	88	33	0	0	2
	CF1	97	98	34	78	73	65	8	9	50	36	72	12	0	0	0
	CF2	95	97	21	83	68	86	24	18	64	55	86	44	0	2	0
	CF3	95	98	34	88	10	86	16	17	79	48	91	59	1	0	0
Water-Jugs	**baseline**	89	94	25	61	82	100	80	25	96	88	100	41	0	0	0
	CF1	44	58	2	45	59	41	50	11	31	56	37	17	0	1	0
	CF2	59	55	0	53	56	43	55	18	37	41	48	10	0	0	0
Hanoi	**baseline**	67	83	20	90	37	83	0	50	60	43	13	23	3	3	0
	CF1	87	77	7	73	20	37	0	0	77	37	3	20	0	0	0
	CF2	50	40	0	10	0	3	0	0	23	30	0	0	0	0	0
Checker-Jumping	**baseline**	72	32	0	0	0	0	0	0	12	8	4	0	0	0	0
	CF1	37	13	0	2	0	0	0	0	0	0	0	0	0	0	0
	CF2	72	20	0	8	0	0	0	0	8	0	4	0	0	0	0
	CF3	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0
River-Crossing	**baseline**	100	75	0	10	0	20	10	0	5	5	65	0	0	0	0
	CF1	95	60	5	30	0	40	15	0	45	15	60	0	0	0	0
Overall (all 30 cells)		**80**	**75**	**20**	**49**	**43**	**53**	**36**	**10**	**54**	**42**	**56**	**15**	**1**	**1**	**0**

Table 7: Exact match accuracy (%) per (model, game, variant). Each game block has one baseline row followed by its 1 to 4 counterfactual variants. Cell color is a per-game blue heatmap. Each game is normalized against *its own maximum* so that within-game contrasts are immediately visible. Opus = Claude-Opus-4.7, Sonnet = Claude-Sonnet-4.6, DSeek-V3 = DeepSeek-V3.2, DSeek-R1 = DeepSeek-R1, Lla-3.3 = Llama-3.3-70B, OLMo32 = OLMo-3.1-32B, Lla-3.1 = Llama-3.1-8B, Qwen3.5 = Qwen3.5-9B.

CF). OLMo-2-7B stays at $\leq 2\%$ everywhere.

D.2 Sudoku partial correctness

We flag one game-specific result that is invisible in Table 7's exact match column. Until the Claude family, end-to-end accuracy on every Sudoku variant was at the floor for all but two models (Sonnet 54%, GPT-OSS-120B 51%, GPT-OSS-20B 42% on the standard 9×9, everyone else $\leq 6\%$). Opus changes the picture by clearing 82% on baseline and 88–89% on three of the four 9×9 CFs, while still scoring 0% on the hex CF. For the remaining models, cell level partial credit (Figure 9) recovers a meaningful ranking. GPT-OSS-120B matches 43 to 57% of cells on every 9×9 variant and 38% on the 16×16 hex CF. DeepSeek-V3.2 reaches 43 to 46% on the 9×9 variants and tops GPT-OSS-120B on the 11-19 CF. OLMo-3.1-32B reaches 18 to 24%, the best among the smaller open weight models. Sudoku was the only game in our suite where exact match was uninformative across most of the model suite. With Opus included, the hex CF remains the lone variant where no model registers above 0%.

Exact-match accuracy on Sudoku is misleading because the global success rate is near zero for every model, on both baseline and CF. Yet the responses are not uniformly empty: many models output an $N \times N$ grid that satisfies a subset of constraints. Figure 9 and 10 show the fraction of cells that match the canonical solution, by model and variant. The exact-match ranking is uninformative (everybody $\leq$6%) but the cell-level ranking is not: GPT-OSS-120B reaches 43-57% correct cells on every 9×9 variant, DeepSeek-V3.2 reaches 43-46% across the 9×9 variants (and is the only model to top GPT-OSS-120B on a 9×9 CF, the 11–19 variant, 46% vs 43%), Qwen3-235B reaches 43% on shifted-boxes, and OLMo-3.1-32B reaches 18-24% on the four 9×9 variants (the highest among the smaller open-weight tier).

D.3 Sudoku as an end-to-end ceiling

D.4 Token usage

As an auxiliary diagnostic of response length, we record **mean total output tokens** per

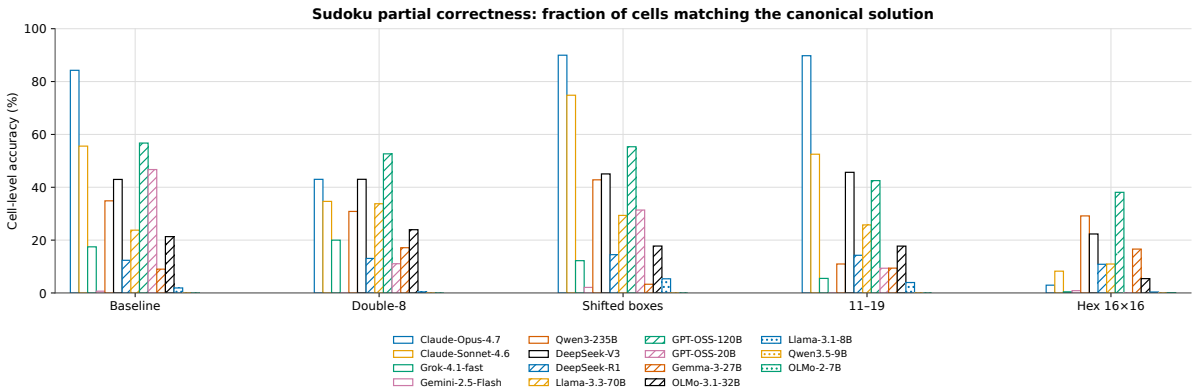

Figure 9: Sudoku cell level accuracy. Each bar is the mean fraction of cells (of $N \times N$) that match the canonical solution for the 100 puzzles in that variant. Exact match accuracy is at or below 1% for all models on all CFs and is omitted. Partial credit reveals a meaningful ranking.

Model	Baseline	CF	ρ
Llama-3.3-70B	924	1028	$+0.58$
OLMo-3.1-32B	268	316	$+0.47$
Llama-3.1-8B	1110	1201	$+0.09$
Qwen3.5-9B	1028	1086	$+0.02$
GPT-OSS-20B	2478	2512	$+0.00$
Gemini-2.5-Flash	217	237	-0.27
Grok-4.1-fast	660	709	-0.29
GPT-OSS-120B	2023	1384	-0.30
OLMo-2-7B	33	42	-0.31
Qwen3-235B	2226	2910	-0.35
Claude-Opus-4.7	4220	8003	-0.45
DeepSeek-V3.2	1104	1602	-0.66
Claude-Sonnet-4.6	**6677**	**8836**	-0.67
DeepSeek-R1	6467	7363	-0.67
Gemma-3-27B	1747	2592	-0.89

Table 8: For each model, **Baseline** and **CF** report mean generated output tokens on baseline and CF evaluation, respectively. ρ is the Spearman correlation between mean output length and exact-match accuracy across the 30 (game, variant) cells.

model and (game, variant) cell. For models with a hidden reasoning channel (Claude-Opus-4.7, Claude-Sonnet-4.6, DeepSeek-R1, GPT-OSS-120B, GPT-OSS-20B) we use the provider-billed `completion_tokens`, which includes thinking tokens. For the remaining models, whose visible response is the complete output, we approximate as character-count / 4 on the raw text.

Table 8 compares mean output length per model on baseline vs CF items. Among models without a hidden reasoning channel, two highly verbose ones, Qwen3-235B ($\sim$2,570 tokens) and Gemma-3-27B ($\sim$2,170 tokens), bracket the high end. OLMo-2-7B is the most terse (around 40 tokens on average). GPT-OSS-20B and GPT-OSS-120B, once their hidden reasoning is included via billed `completion_tokens`, average $\sim$2,500 and $\sim$1,700 tokens respectively. their visible responses were short (≤ 100 tokens), but the bulk of their output is hidden chain of thought. The Claude models and DeepSeek-R1 sit at the top end ($\sim$7–9k billed tokens per call), since their billed total includes the hidden thinking channel.

Per-model Spearman correlations between cell-level mean tokens and cell-level accuracy across all 30 (game, variant) cells (Table 8) show eight of fifteen models with weak or zero correlation ($|\rho| < 0.4$), two non-reasoning open-weight models with positive correlations (Llama-3.3-70B $\rho = +0.58$, OLMo-3.1-32B $\rho = +0.47$), one moderately negative (Claude-Opus-4.7 $\rho = -0.45$), and *four with strong negative correlations* ($\rho \leq -0.66$): Gemma-3-27B (-0.89), Claude-Sonnet-4.6 (-0.67), DeepSeek-R1 (-0.67), DeepSeek-V3.2 (-0.66).

Table 11 reports the mean *total* output length per item for every (model, game, variant) cell, the same 30-cell decomposition used in Table 7. For models that expose a hidden reasoning channel (Claude-Opus-4.7, Claude-Sonnet-4.6, DeepSeek-R1, GPT-OSS-120B, GPT-OSS-20B) we use the provider-billed `completion_tokens`, which counts thinking tokens not present in the visible response. For the remaining models, whose visible response is the complete output, we approximate as character-count / 4 on the raw text. Reading down a game block shows how each model's verbosity changes when a CF rule is applied to that game. reading across a row shows which models inflate their output on the same problem. The scatter underlying the Spearman correlations of Table 8 is reproduced as Figure 10.

Three patterns are visible. (i) Models that scale

verbosity with CF difficulty: Gemma-3-27B and DeepSeek-V3.2 both roughly triple their output on the hardest CFs (Gemma cryptarithmetic baseline 752 $\rightarrow$ base-9 mixed 4,336. DeepSeek sudoku hex 2,181, the only cell where it falls below baseline volume). The two GPT-OSS models behave similarly once their hidden reasoning is included (GPT-OSS-20B cryptarithmetic baseline $\sim$1,500 $\rightarrow$ base-9 mixed $\sim$5,700. GPT-OSS-120B Sudoku baseline $\sim$7,000 with most other cells in the 300–3,000 range). (ii) Models with near-constant output: OLMo-2-7B (8-200 tokens, mostly below 30), Llama-3.1-8B / Qwen3.5-9B (around 1,000-1,500 tokens regardless of difficulty, their context-budget defaults rather than reasoning effort). (iii) Sudoku hex 16$\times$16 generally *lowers* output for the verbose models (because the model gives up earlier on a clearly intractable grid) while *raising* it for the terse models (because they at least attempt to write out the 16×16 grid).

E Model Identifiers

The fifteen models used in the evaluation, with a short note on each (vendor, size, reasoning style) and the exact API identifier:

- **Claude-Opus-4.7**, Anthropic's flagship reasoning model. run with the extended-thinking channel enabled (budget 2048 tokens).
- **Claude-Sonnet-4.6**, Anthropic's mid-tier reasoning model. run with the extended-thinking channel enabled.
- **DeepSeek-R1**, DeepSeek's dedicated reasoning model with a visible chain of thought.
- **Grok-4.1-fast (non-reasoning)**, xAI's fast variant without an explicit reasoning trace.
- **Gemini-2.5-Flash**, Google's fast reasoning model. run with the hidden thinking channel disabled.
- **Qwen3-235B-A22B-Instruct-2507**, 235B-parameter MoE.
- **DeepSeek-V3.2**, DeepSeek's hybrid MoE. run with `reasoning_effort=low`.
- **Llama-3.3-70B-Instruct**.
- **GPT-OSS-120B / GPT-OSS-20B**, OpenAI's open-weight family, run with `reasoning_effort=low`.
- **Gemma-3-27B-IT**, Google's open-weight 27B model (Vertex `gemma-3-27b-it-maas` endpoint).
- **OLMo-3.1-32B-Instruct**, AI2's mid-size open 32B model.
- **Llama-3.1-8B-Instruct**, smaller Llama anchor.
- **Qwen3.5-9B**, smaller Qwen anchor.
- **OLMo-2-7B-Instruct**, AI2's open 7B model, the smallest in the set.

F Reasoning configuration

Seven of the fifteen models expose a controllable hidden-reasoning channel. running them with the channel "on" under Vertex's per-call token cap (max $\sim$8k visible tokens for the GPT-OSS and DeepSeek endpoints) means the analysis trace exhausts the budget before any visible answer is emitted, producing a high rate of empty responses (`finish_reason='length'` for GPT-OSS, `no choices in response` for DeepSeek). To put all models on a comparable visible-tokens basis we use:

- **GPT-OSS-120B**, **GPT-OSS-20B**: `reasoning_effort="low"` in the OpenAI-compatible request body. Vertex's GPT-OSS endpoint accepts only `low/medium/high` (no `minimal`).
- **DeepSeek-V3.2**: `reasoning_effort="low"` via the same extra-body mechanism.
- **Gemini-2.5-Flash**: `thinking_config.thinking_budget=0` via the Vertex Gen AI SDK, disabling the hidden thinking channel entirely.
- **Gemma-3-27B**: no controllable channel. runs at `max_tokens=16384` (raised from the original 2048) to give its visible scratchwork room to terminate in a final grid.

Disabling/dialling-down reasoning consistently increased the rate of parseable responses (e.g. GPT-OSS-120B went from 20% invalid to 7%).

G Base-9 solve strategy: decode-to-decimal vs. native base 9

Measurement details and limitations. The decode rate is a regex based heuristic. For each (model, CF) pair we load every response from the corresponding responses, drop those shorter than 200 characters (short outputs typically contain only the final expression and reveal no strategy), and label the remaining responses as "decoded" if any of the trigger phrases above appears at least once (case insensitive). The numbers reported are the

Display name	API identifier
Claude-Opus-4.7	`claude-opus-4-7`
Claude-Sonnet-4.6	`claude-sonnet-4-6`
DeepSeek-R1	`deepseek-ai/deepseek-r1-maas`
Grok-4.1-fast	`xai/grok-4.1-fast-non-reasoning`
Gemini-2.5-Flash	`gemini-2.5-flash`
Qwen3-235B	`qwen/qwen3-235b-a22b-instruct-2507-maas`
DeepSeek-V3.2	`deepseek-ai/deepseek-v3.2-maas`
Llama-3.3-70B	`meta/llama-3.3-70b-instruct-maas`
GPT-OSS-120B	`openai/gpt-oss-120b-maas`
GPT-OSS-20B	`openai/gpt-oss-20b-maas`
Gemma-3-27B	`google/gemma-3-27b-it-maas`
OLMo-3.1-32B	`allenai/OLMo-3.1-32B-Instruct`
Llama-3.1-8B	`meta-llama/Llama-3.1-8B-Instruct`
Qwen3.5-9B	`Qwen/Qwen3.5-9B`
OLMo-2-7B	`allenai/OLMo-2-1124-7B-Instruct`

Table 9: Model identifiers used in the evaluation.

Var.	Opus	Son.	Grok	Gem.	Qw3	DS-V3	DS-R1	L3.3	G120	G20	Gma	OL32	L3.1	Qw3.5	OL2
Cell-level accuracy (% of cells matching canonical solution)															
base	84	56	17	1	35	43	12	24	57	47	9	21	2	0	0
CF1	43	35	20	0	31	43	13	34	53	11	17	24	0	0	0
CF2	90	75	12	2	43	45	14	29	55	31	3	18	5	0	0
CF3	90	53	6	0	11	46	14	26	43	9	9	18	4	0	0
CF4	3	8	0	1	29	22	11	11	38	0	17	5	0	0	0
Parsed-ok rate (% of responses producing an $N \times N$ integer grid)															
base	85	57	34	1	52	74	19	58	60	47	12	67	6	0	0
CF1	43	41	42	0	51	81	25	86	98	26	27	75	2	0	0
CF2	90	76	27	4	66	90	26	92	100	61	6	64	21	0	0
CF3	90	61	13	0	17	88	24	74	81	35	15	71	18	0	0
CF4	6	18	1	2	66	58	26	95	92	0	35	28	4	0	1

Table 10: Sudoku partial credit. Top half: mean fraction of $N \times N$ cells matching the canonical solution. Bottom half: fraction of responses that produced a well-formed $N \times N$ integer grid (regardless of whether it solves the puzzle). 100 puzzles per (model, variant) cell. Variants: CF1 = double-8, CF2 = shifted boxes, CF3 = 11–19, CF4 = hex 16×16. Model abbreviations: Opus = Claude-Opus-4.7, Son. = Claude-Sonnet-4.6, Gem. = Gemini-2.5-Flash, Qw3 = Qwen3-235B, DS-V3 = DeepSeek-V3.2, DS-R1 = DeepSeek-R1, L3.3 = Llama-3.3-70B, G120/G20 = GPT-OSS-120B/20B, Gma = Gemma-3-27B, OL32 = OLMo-3.1-32B, L3.1 = Llama-3.1-8B, Qw3.5 = Qwen3.5-9B, OL2 = OLMo-2-7B.

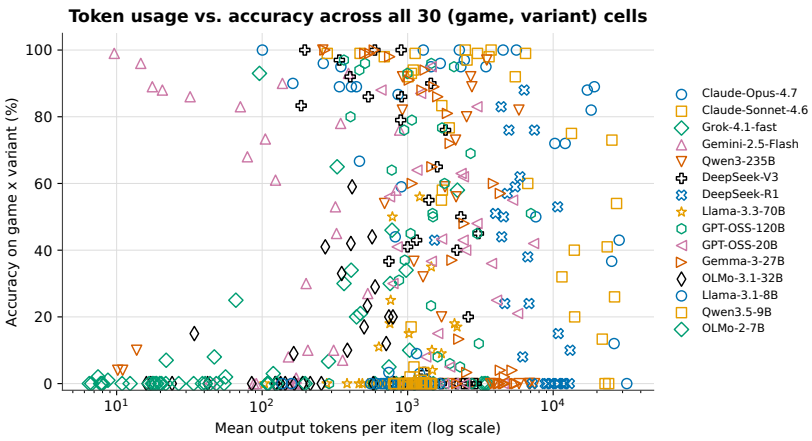

Figure 10: Per (game, variant) cell: mean output tokens (log scale) vs accuracy, colored by model. No clear effort-accuracy slope appears. Gemma-3-27B in particular shows a strong negative trend, consistent with its $\rho = -0.89$ in Table 8.

share of long responses with this label. The heuristic has three known limitations. (i) False positives: a model can mention "base 10" in passing without actually decoding the puzzle. Manual spot checks

Game	Var.	Opus	Son.	Grok	Gem.	Qw3	DS-V3	DS-R1	L3.3	G120	G20	Gma	OL32	L3.1	Qw3.5	OL2
Math-driven games																
Sud.	base	18244	27270	913	91	6913	2400	9085	678	7029	7333	4623	124	1198	820	111
	CF1	28418	26410	834	155	7455	2956	10661	534	1524	4532	4539	85	1456	930	109
	CF2	19190	25333	595	173	6898	2564	10839	463	288	165	4634	144	1291	1109	109
	CF3	17167	23511	732	88	6804	2794	8826	621	1949	2462	4283	123	1519	800	126
	CF4	32056	22720	191	170	2019	2181	11643	678	578	44	2180	255	1062	809	198
Crypt.	base	1281	2490	781	880	2713	1443	4390	912	1454	1452	752	24	1350	1056	18
	CF1	1676	3018	758	831	2763	1820	4966	871	2082	2992	1317	42	1284	1109	19
	CF2	2265	3746	2198	532	3497	2321	7432	890	2708	2504	3888	17	1265	1105	19
	CF3	4528	6459	1032	357	5816	3062	10735	735	3005	5693	4336	16	1664	1153	18
Countd.	base	163	471	22	15	10	339	4835	1466	466	2415	550	407	1345	1021	8
	CF1	263	1125	328	392	923	595	4652	1455	997	2327	508	165	1206	1219	9
	CF2	3452	5491	119	79	1276	1399	6824	1482	1482	4083	1877	743	1231	998	8
	CF3	444	1061	47	32	11	896	4430	1109	946	2403	1517	598	1268	966	8
	CF4	100	280	96	10	261	196	1530	1209	365	1228	270	34	1151	1083	11
G-24	base	411	1006	365	18	694	403	4558	1021	405	1158	1256	352	1301	952	7
	CF1	2451	3626	976	346	2135	1597	6172	1706	1494	3768	1960	713	1528	1094	6
	CF2	1446	2560	474	71	2542	909	6691	2099	777	5262	1610	570	1399	1039	7
	CF3	348	691	409	21	14	537	4014	2185	1069	2994	1026	415	1353	941	8
W-Jugs	base	340	1094	66	124	920	902	6319	764	509	655	590	272	772	1071	20
	CF1	822	1732	56	325	1806	999	5481	634	873	751	2016	502	796	1242	12
	CF2	904	1706	34	319	2153	1152	5914	755	955	846	2443	385	860	1241	17
Non-math games																
Hanoi	base	467	1717	445	138	1104	185	7182	786	1793	1711	2247	529	743	1291	39
	CF1	860	1932	286	105	1710	744	9635	994	1723	1455	2548	792	890	1332	41
	CF2	7598	13975	1114	310	5824	1299	9648	906	1448	792	4191	933	616	1416	44
Check.	base	10271	11492	3231	155	1762	1134	12925	489	3086	1331	5218	167	659	1251	20
	CF1	25244	21753	3354	171	2828	1505	11771	104	2767	2061	5452	214	556	1352	27
	CF2	12095	14050	1501	151	3723	1703	12023	383	1607	969	4333	134	685	1428	20
	CF3	26235	24064	3436	128	1363	1287	11885	278	1839	1769	5592	190	736	1310	22
River	base	5589	13359	3545	206	3896	2608	13070	567	2292	1946	1468	806	159	1464	48
	CF1	2320	6690	744	200	2454	2181	10844	872	1047	1609	1076	580	137	1444	55

Table 11: Mean *total* output tokens per item for every (model, game, variant) cell, including hidden reasoning where present. For Claude (Opus, Sonnet), DeepSeek-R1, and the two GPT-OSS variants we use the provider-billed `completion_tokens` (which counts thinking/reasoning tokens not emitted in the visible response). For every other model, whose CSV has no usage field and whose visible response is the complete output, we approximate as character-count / 4 on the raw response text. Game abbreviations: Sud. = Sudoku, Crypt. = Cryptarithmetic, Countd. = Countdown, G-24 = Game-24, W-Jugs = Water-Jugs, Check. = Checker-Jumping, River = River-Crossing. CF1–CF4 follow the order in Table 1. Model abbreviations: Opus = Claude-Opus-4.7, Son. = Claude-Sonnet-4.6, Gem. = Gemini-2.5-Flash, Qw3 = Qwen3-235B, DS-V3 = DeepSeek-V3.2, DS-R1 = DeepSeek-R1, L3.3 = Llama-3.3-70B, G120/G20 = GPT-OSS-120B/20B, Gma = Gemma-3-27B, OL32 = OLMo-3.1-32B, L3.1 = Llama-3.1-8B, Qw3.5 = Qwen3.5-9B, OL2 = OLMo-2-7B.

of around 30 responses per model on Countdown CF1 found no such cases, but the false positive rate is likely higher on the Cryptarithmetic CFs where the word "decimal" sometimes appears in throw-away clarifications. (ii) False negatives: a model can decode silently by writing only the converted numerals (e.g., "$5 \to 5$, $10 \to 9$, $4 \to 4$") without any of the trigger words.

Five of the 22 CF combinations alter the numeric base: Countdown CF1 (base-9 source numbers and target), Game-24 CF1 (base-9 operands and target), and Cryptarithmetic CF1–CF3 (base-9 addition / subtraction / mixed). When we examined chain-of-thought responses on these combinations, a clear pattern emerged: models often do not solve in base 9 at all. Instead they decode the base-9 numerals into base 10, solve the standard problem in decimal, and re-encode the answer back to base 9. Below is a representative DeepSeek-V3.2 transcript on Countdown CF1:

> ***Step 1: Decode the base-9 numbers to base 10.*** $5_9 = 5_{10}$, $10_9 = 9_{10}$, $4_9 = 4_{10}$, *target* $125_9 = 81+18+5 = 104_{10}$. ***Step 2: Find an expression in base-10 that reaches 104.*** $(9+4) \times (9-1) = 104$. ***Step 3: Re-encode each literal back to base 9.***

Table 12 measures how prevalent this approach is across combinations and models, by counting how many responses (> 200 characters of chain-of-thought) contain at least one explicit decode-to-decimal phrase (variants of "base 10", "decimal", "decode", "convert to base 10", or a LaTeX subscript _10). Sub-10B models and GPT-OSS-* are omitted because their responses are too short to

Model	C-dn	G-24	Cr +	Cr −	Cr ±
Opus	—	50	0	12	80
Sonnet	100	100	13	0	45
Grok	100	100	23	46	24
Gemini	100	100	51	2	3
Qwen3	100	100	95	97	76
DS-V3	100	100	74	54	72
DS-R1	52	71	4	6	6
Lla-3.3	100	100	45	33	16
Gemma	100	100	70	51	68

Table 12: % of long-form responses ($>$ 200 chars) that explicitly decode base-9 numerals to base 10 before solving. C-dn = Countdown CF1, G-24 = Game-24 CF1, Cr +/−/± = Cryptarithmetic base-9 add / sub / mixed. Model abbreviations follow Table 11. Claude-Opus-4.7's Countdown CF1 cell is "—" because Opus performs its chain-of-thought in a hidden thinking channel and the visible response on Countdown is only the final expression ($\approx$ 26 chars on average). The judge has nothing to classify. Opus's other base-9 figures are computed from a 30-sample judge pass on the visible response.

classify reliably (most are below 200 characters.).

On Countdown and Game-24 every model with visible reasoning decodes to base 10 *universally* (100% of responses on both). These two CFs operationally test whether a model can do a base-9 $\rightleftharpoons$ base-10 conversion in addition to the standard game. the underlying "how do I reach the target" search runs in decimal. This is consistent with the relatively high accuracy of these combinations in Table 7 (Gemma 99% on Countdown CF1. Gemini 78% on Game-24 CF1).

On Cryptarithmetic the picture diverges by model. Two reasoning models, Gemini-2.5-Flash and Grok-4.1-fast, mostly stay native in base 9 (2–46% decode rate), in the column-arithmetic structure of cryptarithmetic, working in the puzzle's own base is just as natural as converting. Qwen3-235B, DeepSeek-V3.2 and Gemma-3-27B prefer to decode and verify in decimal (51–97%). Cryptarithmetic CF1 (base-9 add) shows the same accuracy ceiling either way, both Gemini (51% decode, 58% accuracy) and Qwen3-235B (95% decode, 89% accuracy) do well, but on the harder CF3 (base-9 mixed) the decode route is no help: Gemini stays native and scores 7%, Qwen3 decodes and scores 82%, DeepSeek decodes and scores 45%. The decode-route advantage on cryptarithmetic mixed is consistent with the observation that base-9 chained $+/-$ requires *simultaneous* carry and borrow tracking within the same column chain, which is exactly what makes mixed cryptarithmetic harder than pure addition or subtraction *in base 9*, and exactly what decoding to decimal removes.

H Non-math partial correctness

For the three non-math games (Hanoi, Checker-Jumping, River-Crossing) the verifier replays moves move-by-move against the variant's rule set. Each response falls into one of four buckets:

- **correct**: every move legal AND the play reaches the goal state.
- **legal, no goal**: moves parse and every move is legal under the variant, but the sequence does not reach the goal.
- **parsed, illegal**: moves parse but the verifier rejects at least one move as illegal under the variant rule.
- **unparseable**: no extractable move sequence in the response.

Table 13 shows, per (model, game, variant), the rate of each non-correct bucket. The exact-match correct rate appears in Table 7. Three patterns are visible: (i) the *parsed-illegal* bucket dominates for most models on most non-math combinations, models follow the move-list format but break a rule mid-sequence (e.g. in Hanoi-adjacent they make a non-adjacent move, in Checker the slide hits an occupied square). (ii) the *unparseable* bucket is concentrated on Qwen3.5-9B (36-63% on most combinations), Gemini-2.5-Flash on Hanoi-adjacent (60%), Claude-Opus-4.7 on Checker asymmetric and two-empty (57% and 72%, where it produces long reasoning prose without the structured move list our parser expects), and OLMo on three of the four checker variants. (iii) the *legal-no-goal* bucket is rare (mostly $\leq 10\%$), but Gemini reaches 40% on River-Crossing-neutral and 30% on Checker asymmetric, it produces fully legal but incomplete plays.

I Response-style fingerprints

The causal analysis of §5.2 focuses on whether a CF failure is caused by (a) not registering the rule or (b) registering it but reverting to the textbook procedure. Independently of that, the way each model formats and packages its response carries its own signal. Patterns that distinguish vendors at fixed accuracy without speaking to causation. We report four such patterns, each split by baseline (B) and counterfactual (CF) over the full 30 baseline+CF cells. "Others $\leq x\%$" lumps every model whose row stays under x in both buckets.

Game	Var.	Opus	Son.	Grok	Gem.	Qw3	DS-V3	DS-R1	L3.3	G120	G20	Gma	OL32	L3.1	Qw3.5	OL2
Legal but no goal (%)																
Hanoi	base	0	0	0	10	0	13	0	10	10	3	0	7	17	20	0
	CF1	0	0	7	20	0	7	0	0	3	37	0	3	7	33	0
	CF2	10	17	0	67	0	0	0	0	47	40	3	3	3	73	0
Check.	base	4	8	0	20	0	4	0	0	0	4	0	4	4	20	0
	CF1	57	45	0	37	0	0	0	0	7	17	0	2	0	62	2
	CF2	16	20	0	0	0	0	0	0	4	8	0	0	0	36	0
	CF3	72	68	0	8	0	0	0	0	12	16	0	8	0	40	0
River	base	0	15	0	85	0	15	0	0	40	10	0	5	0	85	0
	CF1	0	5	0	65	0	15	0	0	15	15	5	5	5	85	0
Parsed, but at least one illegal move (%)																
Hanoi	base	33	17	80	0	63	3	93	40	30	50	87	70	80	77	100
	CF1	13	23	87	7	80	57	93	100	20	27	97	77	93	67	100
	CF2	33	43	100	23	100	97	53	100	30	17	97	97	97	27	100
Check.	base	24	60	100	80	100	88	68	100	88	88	96	96	96	80	100
	CF1	7	42	93	62	100	100	52	100	93	80	92	98	100	38	98
	CF2	12	60	100	92	100	100	68	100	88	92	96	100	100	64	100
	CF3	16	32	96	92	100	100	52	100	88	84	100	92	100	60	100
River	base	0	10	80	5	100	65	65	100	55	75	35	95	100	15	100
	CF1	5	35	85	5	100	45	65	100	40	65	35	95	95	15	100
Unparseable (%)																
Hanoi	base	0	0	0	0	0	0	7	0	10	3	0	0	7	13	0
	CF1	0	0	0	17	0	0	7	0	0	27	0	3	7	30	0
	CF2	7	17	0	60	0	0	47	0	47	53	0	3	3	63	0
Check.	base	4	8	0	0	0	8	32	0	0	4	0	4	4	12	0
	CF1	57	45	7	7	0	0	48	0	7	20	8	0	0	53	0
	CF2	16	16	0	0	0	0	32	0	4	8	0	0	0	36	0
	CF3	72	68	4	0	0	0	48	0	12	16	0	0	0	12	0
River	base	0	15	20	65	0	15	25	0	30	20	0	5	0	70	0
	CF1	0	5	10	25	0	0	20	0	15	15	0	0	5	80	0

Table 13: Non-math partial-credit breakdown per (model, game, variant). The three rates (legal-but-no-goal, parsed-illegal, unparseable) plus the correct rate from Table 7 sum to 100 in each cell (modulo rounding). Non-math games use temperature 0.7 and 5 runs per puzzle, so each cell is computed over $5 \times |\text{puzzles}|$ rows. Game abbreviations: Check. = Checker-Jumping, River = River-Crossing. Variants: Hanoi CF1 = four-pegs, CF2 = adjacent. Checker CF1 = asymmetric, CF2 = jump-two, CF3 = two-empty. River CF1 = neutral. Model abbreviations: Opus = Claude-Opus-4.7, Son. = Claude-Sonnet-4.6, Gem. = Gemini-2.5-Flash, Qw3 = Qwen3-235B, DS-V3 = DeepSeek-V3.2, DS-R1 = DeepSeek-R1, L3.3 = Llama-3.3-70B, G120/G20 = GPT-OSS-120B/20B, Gma = Gemma-3-27B, OL32 = OLMo-3.1-32B, L3.1 = Llama-3.1-8B, Qw3.5 = Qwen3.5-9B, OL2 = OLMo-2-7B.

Self-revision: multiple `<answer>` blocks per response. Some models emit more than one `<answer>...</answer>` block, effectively rewriting their answer mid-output. We measure the share of responses with ≥ 2 such blocks.

Model	B	CF	Model	B / CF
DS-R1	11.9	14.6	Gemma	0.0 / 3.7
Opus	12.5	2.2	DS-V3	0.8 / 4.2
Qwen3	2.8	7.2	Others ($\leq 0.6\%$)	—

For Opus and DeepSeek-R1 the revision is *productive*. Grading by the last block instead of the first recovers $103/112$ math-game responses for Opus and $+201/478$ for DeepSeek-R1. The first block is essentially a discarded draft. For Qwen3-235B the revision recovers only $+15/221$, and for DeepSeek-V3.2 and Gemma it recovers *zero*: the model rewrites but does not improve. Self-revision is therefore a marker of a specific kind of internal critic, not just verbosity. CF inflates the rate for the open-weight reasoners (Gemma $0 \rightarrow 3.7\%$, V3 $0.8 \rightarrow 4.2\%$, Qwen3 $2.8 \rightarrow 7.2\%$). Opus decreases from B to CF because most of its multi-block baselines are the hard Game-24 / Sudoku problems where it agonizes.

Refusal: "I cannot ..." / "unable to ..." / "not possible".

Model	B	CF	Model	B	CF
DS-R1	67.5	57.5	Qwen3	24.4	22.0
DS-V3	20.7	23.1	Gemma	10.4	15.6
Lla-3.3	11.9	6.8	Grok	2.6	3.4
Gemini	3.3	2.3	Opus/Son/GPT-OSS	≤ 0.2	≤ 0.7

DeepSeek-R1 refuses on more than half of its prompts, including baselines (67.5% vs 57.5% CF). Difficulty triggers refusal, not the rule change. The two Claude models and GPT-OSS-* almost never refuse ($\leq 0.7\%$). The split is consistent with vendor-level differences in how models are trained to handle inputs that look unsolvable: "try anyway" (Anthropic, OpenAI's GPT-OSS) and "opt out" (DeepSeek) are both available training targets.

Confident format errors: `extracted_answer` present but `valid=False`.

Model	B	CF	Model	B	CF
Gemini	1.9	**19.5**	Gemma	9.0	11.7
Qwen3	3.8	5.5	Grok	2.4	2.6
DS-V3	1.3	2.4	DS-R1	0.7	1.2
Lla-3.3	4.0	2.4	Opus/Son/GPT-OSS	≤ 0.7	≤ 1.5

Gemini-2.5-Flash is the only model with a strong CF-specific signal: malformed-but-non-empty rates jump $10\times$ from B to CF (1.9 to 19.5%). The Sudoku 11-19 and hex variants drive most of the inflation: Gemini outputs a 9×9 array that "looks right" but contains the digit 1 instead of 11, or a 16×16 array with a hex-letter typo. The Claudes are below 1% on both sides.

Verification language: "verify", "double-check", "sanity check".

Model	B	CF	Model	B	CF
Sonnet	29.9	42.6	DS-V3	35.2	38.2
Opus	3.8	14.0	DS-R1	58.7	50.3
Qwen3	56.2	55.4	Gemma	85.6	75.7
Grok	25.0	26.6	GPT-OSS-*	0.0	0.0

Sonnet and Opus reach for verification language $\sim 3\times$ more often on CFs than on baselines (Sonnet $30 \rightarrow 43\%$, Opus $4 \rightarrow 14\%$). On Opus's hardest cells the rate reaches 84% (cryptarithmetic base-9 mixed), 53% (base-9 sub), 49% (base-9 add). Gemma moves the other way – when it is confused it omits the verification step rather than running it. This pattern correlates with the *register-and-execute* regime of §5.2: the models that benefit most from rule acknowledgement are also the ones that most consistently verify their answers.

J Example prompts

One real baseline instance per game, taken verbatim from the released CSVs (the first row, `*_baseline_0001`). Counterfactual prompts share the same template. Only the rule clause changes.

Sudoku (baseline).

```
Solve the following standard Sudoku
puzzle.
Rules: Fill the 9×9 grid so that each
row, column, and 3×3 box contains each
of the digits 1 through 9 exactly once.

Puzzle (. = empty cell):
4 9 . | . . . | 6 . 7
3 6 5 | . . 4 | . 8 .
. 2 7 | . . . | . . .
————————————
. . 6 | . . 3 | . 7 9
. 8 . | . 9 . | . 4 .
. 4 . | 6 8 . | . 3 .
————————————
8 7 . | . . 6 | . . .
. . . | . 1 2 | . . .
. 1 . | . . . | . 2 5

Provide the completed 9×9 grid as a 2D
JSON array (9 lists of 9 integers).
```

Cryptarithmetic (baseline).

```
Solve the following cryptarithmetic
puzzle using standard arithmetic (Base
10).
Rules:
  - Valid digits: {0, 1, 2, 3, 4, 5, 6,
7, 8, 9}.
  - Each letter represents a unique
digit.
  - No multi-letter word may begin with
0.
  - Addition carries when a column sum
reaches 10 (standard rule).
  - Subtraction borrows by adding 10 to
the current digit (standard rule).

Equation: GFCD + FEC = BAEAF

Provide the letter-to-digit assignment
as a JSON object, e.g. {"A": 1, "B": 5,
"C": 3}.
```

Countdown (baseline).

```
Solve the following Countdown-style
puzzle.
Rules:
  - You are given six numbers and a
single integer target.
  - Use a subset of the numbers (each at
most once).
  - Combine them with +, −, ×, ÷ and
parentheses.
  - Every intermediate division must
yield an exact integer (no fractions).

Numbers: 7, 6, 1, 8, 6, 3
Target: 298

Provide only the arithmetic expression
(you may use ×, ÷, − or equivalently
*, /, -; no equals sign).
```

Game-24 (baseline).

```
Solve the following Game of 24 puzzle.
Rules:
  - You are given four positive
integers.
  - Use each number exactly once.
  - Combine them with +, −, ×, ÷ and
parentheses.
  - Every intermediate division must
yield an exact integer.
  - The expression must evaluate to 24.

Numbers: 11, 2, 1, 12

Provide only the arithmetic expression
(you may use ×, ÷, − or equivalently
*, /, -; no equals sign).
```

Water-Jugs (baseline).

```
You are playing the classic Water Jug
puzzle.
Rules − allowed moves:
```

- Fill a jug from the tap (fills it to full capacity).
- Empty a jug (drains it completely to 0).
- Pour from one jug to the other (transfers water exactly, no loss).

Goal: reach the exact target volume in either jug using the fewest moves.

Puzzle: Jug A capacity: 7L | Jug B capacity: 8L | Starting state: [0, 0] | Target: 2L in either jug

Provide the solution path as a JSON list of [Jug_A, Jug_B] states from start to goal, e.g. [[0,0],[5,0],[5,3]].

Tower of Hanoi (baseline).

You are solving the Tower of Hanoi puzzle.

RULES:
- There are 3 pegs labeled A, B, C.
- There are 2 disks numbered 1 (smallest) to 2 (largest).
- Only the top disk of a peg may be moved.
- A larger disk may never be placed on top of a smaller disk.
- Move ALL disks from peg A to peg C.

INITIAL STATE:
Peg A: 2, 1 [bottom → top]
Peg B: (empty)
Peg C: (empty)

GOAL STATE:
Peg A: (empty)
Peg B: (empty)
Peg C: 2, 1 [bottom → top]

Output format: one move per line, "disk ⟨N⟩ : ⟨SRC⟩ -> ⟨DST⟩". Think step by step, then output all moves.

Checker-Jumping (baseline).

You are solving the Checker Jumping (Frog Puzzle).

SETUP:
- The board has 5 positions numbered 0 to 4 (left to right).
- 2 Red (R) frogs start on the LEFT. 2 Blue (B) frogs start on the RIGHT.
- There is 1 empty space in the middle.

RULES:
- Red frogs can only move to the RIGHT.
- Blue frogs can only move to the LEFT.
- SLIDE: move 1 position into an immediately adjacent empty space.
- JUMP: leap over exactly 1 adjacent opposite-color frog into the empty space beyond.

INITIAL BOARD:
[0]=R [1]=R [2]=_ [3]=B [4]=B

GOAL BOARD:
[0]=B [1]=B [2]=_ [3]=R [4]=R

Output format: one move per line, "⟨COLOR⟩ ⟨FROM⟩ -> ⟨TO⟩". Think step by step, then output all moves.

River-Crossing (baseline).

You are solving the River Crossing puzzle.

SETUP:
- There are 2 agent-actor pairs: (A_1, a_1), (A_2, a_2).
- Each pair consists of one AGENT (A_i) and one ACTOR (a_i).
- Everyone starts on the LEFT bank.

Goal: move everyone to the RIGHT bank.
- Boat capacity: 2 people per trip (minimum 1; boat cannot travel empty).

SAFETY RULE:
- On any bank, if any ACTOR (a_i) is present alongside any AGENT (A_j), then EVERY actor on that bank must have their OWN agent present.
- A bank with only agents, only actors, or nobody is always safe.

TRIP DIRECTION: odd trips go LEFT → RIGHT; even trips go RIGHT → LEFT.

INITIAL STATE: Left bank: A_1, A_2, a_1, a_2 | Right bank: (empty)
GOAL STATE: Left bank: (empty) | Right bank: A_1, A_2, a_1, a_2

Output format: one trip per line, "trip ⟨N⟩ : ⟨person1⟩[, ⟨person2⟩]". Verify safety after each trip.