BEYOND ACCURACY: DECOMPOSING THE REASONING EFFICIENCY OF LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Accuracy per token summarizes reasoning efficiency but does not reveal whether differences arise from failed attempts, costly successful answers, or disproportionately long reasoning relative to the effort a problem demands. We introduce a two-layer protocol that connects these distinctions to the same efficiency score. The outcome layer separates answer availability, correctness among available answers, and generation cost. The workload layer relates cost to task structure, separating mean generated tokens per workload unit from their association with workload. Across 785,541 responses on twelve benchmarks, cost gaps persist among successful answers: Qwen3.5-27B generates 1.79–3.46 times as many tokens as Gemma-4-31B for correct outputs on shared problems across ten benchmarks. Relative costs also change with workload. On ZebraLogic, Qwen3.5-27B's cost relative to GPT-OSS-120B changes from 4.49 at low workload to 0.78 at high workload, despite similarly long responses on average. Across nine checkpoints on four annotated benchmarks, simultaneous 95% confidence intervals support relative-cost changes in 82 of 144 model-pair comparisons and effort-induced changes in 21 of 36 within-checkpoint comparisons. These results show that successful-answer cost and workload response are complementary properties of reasoning models: cost gaps can persist on shared successful questions while changing substantially with task structure and inference settings.

1 INTRODUCTION

Reasoning models differ not only in whether they solve a problem, but also in how many tokens they generate before answering. Longer chains of thought, search, and self-verification can improve accuracy while increasing token cost (Snell et al., 2025; Muennighoff et al., 2025; Guo et al., 2025). Accuracy per generated token summarizes this tradeoff, but the same score can accompany different behaviors: frequent failures to return a scorable answer, delivering incorrect answers, or costly correct answers. Average output length leaves a second question unresolved: is one model consistently more expensive, or does the cost gap change with task structure?

We address these ambiguities through three diagnostic questions. **Answer outcomes:** is an efficiency gap driven by unavailable answers, incorrect answers, or greater generation cost? **Successful-answer cost:** does the cost gap persist on the problems that both models answer correctly? **Workload response:** does one model become relatively more expensive as the reasoning workload of problems increases? Together, these comparisons distinguish differences in answer delivery and correctness from differences in the cost of successful responses, and show whether an average cost advantage holds across workload groups.

Our protocol combines exact accounting with comparisons on shared questions. The *outcome layer* separates answer availability, correctness among available answers, and mean generated cost. The *workload layer* uses a benchmark-specific reference, such as reference-solution steps, proof depth, or constraint count, to separate mean tokens per workload unit from their association with workload. The accounting quantifies how these components combine in an efficiency difference. Shared-question comparisons then test whether cost gaps persist in successful answers, while workload-group comparisons test whether relative cost changes with task structure. These complementary analyses use final outcomes, token counts, and workload assignments, without requiring the evaluated models' reasoning traces.

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

Across 785,541 responses on twelve benchmarks, we establish three findings. First, cost gaps persist among correct outputs on problems both models solve. Second, model-wide cost differences coexist with substantial workload-dependent changes, which can reverse which model is cheaper. Third, changing reasoning effort can alter these patterns within a checkpoint, with questions, sampling parameters, and output caps held fixed. Successful-answer comparisons across ten benchmarks establish the first finding; nine-checkpoint comparisons on four annotated benchmarks establish the latter two. Controlled CogniLoad (Kaiser et al., 2026) variation separates changes in task structure, while solver-derived workload extends the analysis to tasks without native workload annotations.

The resulting profiles provide a way to assess model improvements beyond a change in average response length. Shorter average responses can reflect cheaper successful answers, less expenditure on failed responses, or reductions concentrated in particular workload groups. Our protocol identifies which of these changes accompanies an efficiency gain while retaining the full-benchmark accuracy and cost that define it. The resulting contribution is an empirical account of both persistent successful-answer cost differences and the ways generation responds to task structure and inference settings.

2 Related work

Inference-time scaling and overthinking. Inference-time scaling allocates more test-time computation to improve reasoning (Snell et al., 2025; Muennighoff et al., 2025; Guo et al., 2025). Long traces can contain useful intermediate work, but also restatements, hesitation loops, and error-amplifying self-doubt (Yang et al., 2025; Chen et al., 2025; Sui et al., 2025; Wei et al., 2025). Our protocol measures how the resulting gains in correctness and changes in generated cost contribute to token efficiency.

Efficiency-aware evaluation. Budget-aware metrics, accuracy–cost tradeoffs, and length-control methods already evaluate quality jointly with cost (Wang et al., 2024a; Du et al., 2025; Ma et al., 2025). OckBench also relates generated length to problem difficulty, defined through average accuracy across its evaluated models (Du et al., 2025). Our native workload comes from task annotations, independently of the evaluated models' accuracy. Our analysis distinguishes costly successful answers from workload-dependent changes in relative cost. Correct-response comparisons on shared questions test whether failed attempts suffice to explain a cost gap, common workload groups test whether the gap behaves as a fixed multiplier, and within-checkpoint effort comparisons test whether that workload response changes under an inference control. The decomposition quantifies how the associated outcome and cost differences combine in aggregate efficiency. Length-control methods seek to change model behavior (Ma et al., 2025; Aggarwal & Welleck, 2025; Arora & Zanette, 2025); our protocol evaluates which outcomes and workload groups account for the resulting efficiency change.

Process and workload evaluation. Step-level evaluation and process supervision often require reference traces, human labels, reward models, or LLM judges (Lightman et al., 2024; Lee & Hockenmaier, 2025). Post-training methods such as GRPO-λ instead improve credit assignment during learning (Parthasarathi et al., 2025). Our analysis uses final outcomes. Where benchmark structure is available, we use CogniLoad's generator, GSM8K's gold solutions, ProofWriter's proof depth, and ZebraLogic's constraints to define workload (Kaiser et al., 2026; Cobbe et al., 2021; Tafjord et al., 2021; Lin et al., 2025). LiveCodeBench supplies execution-based correctness (Jain et al., 2025), while MathArena supplies external competition-math outputs (Balunović et al., 2025). Together, these benchmarks support evaluation across structured reasoning, executable solutions, and competition problems.

3 Evaluation protocol

We first define and separate the outcomes associated with generated tokens: correct answers, incorrect answers, and answers unavailable to the scorer (outcome layer). We then relate the generated tokens to the workload measure associated with each task (workload layer).

3.1 OUTCOME LAYER: WHERE IS TOKEN EFFICIENCY LOST?

Let a model response to benchmark instance I have correctness $S \in \{0, 1\}$ and generated-token count $T > 0$. Here, expectations, denoted $\mathbb{E}[\cdot]$, are empirical, over benchmark instances and repeated attempts for an instance. We introduce the pooled efficiency E_0, as the correct answers per 1,000 generated tokens:

$$E_0 := \frac{1000\,\mathbb{E}[S]}{\mathbb{E}[T]} = \frac{1000\sum_i S_i}{\sum_i T_i}. \tag{1}$$

The last expression applies when responses have equal weight, as within each reported setting–benchmark evaluation. We also report accuracy and mean generated length alongside E_0 to retain its quality and cost components.

Let $A \in \{0, 1\}$ indicate whether a response contains an *available answer* $A = 1$ or an *unavailable answer* $A = 0$. An *available answer* refers to an answer on an instance which can be scored, while an *unavailable answer* usually results from a truncated response due to reaching the context limit, or an unscorable formatting of it. Since every correct answer is available, we have $S \leq A$. Write $r_a = \mathbb{P}(A = 1)$ for answer availability and $r_c = \mathbb{P}(S = 1 \mid A = 1)$ for correctness among available answers. Then

$$E_0 = 1000\frac{r_a r_c}{\mathbb{E}[T]}, \qquad \Delta \log E_0 = \Delta \log r_a + \Delta \log r_c - \Delta \log \mathbb{E}[T], \tag{2}$$

For positive factors, $\Delta \log x = \log(x/x_{\text{ref}})$ compares a model or setting with a different reference on the same questions. Dividing the efficiencies and taking logarithms gives the additive decomposition in Eq. (2); Appendix A.1 provides the intermediate step. This is an exact accounting identity and requires no independence assumption. Positive contributions w.r.t. the reference increase efficiency and negative contributions reduce it. Effort comparisons use the low reasoning effort setting as the reference.

The decomposition above compares a model with a reference, using outcome rates and token costs computed from their respective responses. We complement this aggregate comparison with three diagnostics: *Where tokens go.* For each evaluation, we report the fraction of responses that are correct, incorrect but available, or unavailable, together with the fraction of generated tokens that each outcome consumes in total. A rare outcome can still account for a large share of cost. *Correctness among shared questions.* Conditional correctness r_c is computed on each model's own available answers, so two models may be scored on different questions: a model that fails to answer its hardest questions can appear more accurate. We therefore also compare correctness on the questions for which both models return at least one available answer, averaging over available attempts within each question and then equally over questions. *Cost of successful answers.* To test whether a cost gap arises only from failed attempts, we consider each pair of models and retain questions on which each model produces at least one correct response. On this shared question set, we compare the models' mean generated lengths in two ways: using only correct responses, and using all responses, including incorrect and unavailable answers. For each model and comparison, we first average the included response lengths within each question and then equally across questions. Comparing the resulting model-to-model cost ratios shows how restricting to correct responses changes the cost gap while holding the question set fixed (Appendix G.1).

3.2 WORKLOAD LAYER: GENERATED LENGTH RELATIVE TO TASK STRUCTURE

In many benchmarks, there are instances that demand different amounts of reasoning effort to solve. We refer to this as the instance implied reasoning *workload*. We assign to each instance a positive workload $W(I)$ in benchmark-specific units, such as reference-solution steps or constraint atoms (more details on this are provided later in this section). The *verbalization overhead* $\mathrm{VO} = T/W$ is the number of tokens generated per workload unit, and the *mean overhead* $\overline{\mathrm{VO}} = \mathbb{E}[T/W]$ averages this normalized length across instances.

Mean tokens per workload unit do not by themselves determine mean token cost: it also matters whether the larger per-unit costs occur on larger-workload instances. Since $T = W\,\mathrm{VO}$,

$$\mathbb{E}[T] = \mathbb{E}[W]\,\overline{\mathrm{VO}}\,\kappa, \qquad \kappa = \frac{\mathbb{E}[T]}{\mathbb{E}[W]\,\overline{\mathrm{VO}}} = 1 + \frac{\mathrm{Cov}(W, \mathrm{VO})}{\mathbb{E}[W]\,\overline{\mathrm{VO}}}. \tag{3}$$

The *workload coupling* κ is greater than one when workload and tokens per unit have positive covariance, less than one when they have negative covariance, and one when the covariance is zero. Multiplying every response length by a constant changes $\overline{\mathrm{VO}}$ but leaves κ unchanged. In other words, for two models with the same $\overline{\mathrm{VO}}$ and different κ values, the model with a larger κ will generate a larger amount of tokens per unit of workload solving high workload instances, relatively to a smaller amounts of tokens per unit of workload for low workload instances. [1]

For models evaluated on identical instances, weights, and workload assignments, $\mathbb{E}[W]$ cancels, and we can replace $\mathbb{E}[T]$ in Eq. (2) with

$$\Delta \log E_0 = \Delta \log r_a + \Delta \log r_c - \Delta \log \overline{\mathrm{VO}} - \Delta \log \kappa. \tag{4}$$

The first two terms describe differences in answer availability and conditional correctness (outcome layer) relative to the reference. The last two account for generation cost through mean overhead and workload coupling. Together, the four signed terms sum exactly to $\log(E_0/E_{0,\mathrm{ref}})$. Exponentiating this sum gives the efficiency of the compared model divided by that of its reference. In some figures we show cross-benchmark summaries that average within-benchmark log contrasts rather than pool incompatible workload units.

Does relative cost change with workload? Coupling accounts for the pooled mean cost. To test directly whether one model becomes relatively more expensive at higher workload, let $\overline{T}_{m,i}$ be model m's mean generated length on question i. Let L contain instances at or below the workload's 25th percentile and H those at or above its 75th percentile, including boundary ties. For models m and n, or two effort settings of one model, compute

$$D_{mn} = \frac{1}{|H|} \sum_{i \in H} \log \frac{\overline{T}_{m,i}}{\overline{T}_{n,i}} - \frac{1}{|L|} \sum_{i \in L} \log \frac{\overline{T}_{m,i}}{\overline{T}_{n,i}}. \tag{5}$$

The quantity $\exp(D_{mn})$ is the high-workload relative cost divided by the low-workload relative cost, using geometric means of within-question cost ratios. A positive D_{mn} means that model m becomes relatively more expensive at high workload and a negative value means the opposite. If the geometric-mean cost ratio is the same in the low- and high-workload groups, then $D_{mn} = 0$. We summarize the magnitude of change by $\exp(|D_{mn}|)$: one indicates no change between the groups, and two indicates a twofold change in relative cost, in either direction. Workload determines the groups, while the ratios use raw lengths: on each shared instance, W_i cancels from the models' normalized-cost ratio. Thus, D_{mn} compares endpoint groups, whereas κ accounts for the full workload distribution. We interpret both alongside accuracy and availability.

Workload definitions. *Native workload* is defined from benchmark annotations or reference solutions before examining the evaluated model's output. CogniLoad counts all condition clauses and the update clauses affecting the queried person. GSM8K uses the number of steps in the reference solution. ProofWriter uses $1 +$ proof depth, retaining positive workload for depth-zero questions. ZebraLogic counts parsed constraint atoms. Every compared model receives the same workload assignment for each question. These benchmark-specific scales are given in Appendix B.

Solver-derived workload extends the analysis when native workload annotations are unavailable by constructing a reference-based workload from successful model outputs. We rank reference models by benchmark-level E_0 and assign each instance as workload the output length of the highest-ranked model that answers it correctly. Controls using native, constant, and permuted workloads, together with reference-family removals, assess this scale (Appendix G.4).

4 Experimental design

Our study combines four experiments (Table 1). A setting specifies a model checkpoint and its reasoning effort configuration, including reasoning effort, sampling parameters, and generation limits.

[1] Consider two equally weighted questions with workloads 1 and 3. Model A generates 50 and 450 tokens, respectively, while model B generates 150 tokens on each question. Their tokens per workload unit are therefore $(50, 150)$ and $(150, 50)$: both average 100. Yet their mean response lengths are 250 and 150 tokens. With mean workload 2, their coupling factors are $250/(2 \cdot 100) = 5/4$ and $150/(2 \cdot 100) = 3/4$, respectively. Model A has greater mean cost because its higher tokens per workload unit occur on the larger-workload question.

Table 1: Evaluation scope and inference budgets. Counts give the number of responses across all settings and attempts. "Settings" denotes 31 configurations on CogniLoad and 14 on each other task in the model comparison; the remaining rows give the number of configurations in each experiment.

Experiment	Settings	Responses	Scope and budget
Model comparison	31 / 14	115,541	Four tasks with model-specific context limits
Reasoning effort	21	597,744	Ten tasks with 81,920 output tokens
Math/code	4	65,536	Two tasks with 16,384 output tokens
Competition math	28	6,720	Two tasks at specified provider settings

The reasoning-effort experiment evaluates nine checkpoints at 21 settings on ten benchmarks: CogniLoad, GSM8K, ProofWriter, ZebraLogic, MATH-500, GPQA, MMLU-Pro, MuSR, ReClor, and LiveCodeBench (Rein et al., 2024; Wang et al., 2024b; Sprague et al., 2024; Yu et al., 2020). Sampling parameters and output caps remain fixed within each checkpoint.

Our primary cross-model analysis uses these nine models with reasoning on (high effort for GPT-OSS): 36 pairs on each of four annotated benchmarks, giving 144 workload comparisons. Within-checkpoint effort comparisons use high versus low for GPT-OSS and on versus off otherwise: 90 comparisons across ten benchmarks, including 36 with native workload annotations.

A broader model-comparison experiment evaluates 31 settings on 1,325 CogniLoad questions and fourteen models on GSM8K, ProofWriter, and ZebraLogic. CogniLoad supports controlled variation of statement count, intrinsic difficulty, and distractor content. Separate experiments cover four settings on MATH-500 and LiveCodeBench, and 28 checkpoints on AIME 2025 and HMMT February 2025.

Generated cost includes every reasoning and final-answer token recorded by the runtime or provider, on successful and unsuccessful responses alike. The two LiveCodeBench experiments use different question sets, with effort comparisons holding questions and output caps fixed within each experiment. Appendix F details generation, scoring, and stopping.

We estimate uncertainty with 999 paired bootstrap resamples of questions, retaining all models and repeated attempts together. Where recorded, ProofWriter parent groups and LiveCodeBench contests are resampled as clusters. Simultaneous 95% intervals cover the workload contrasts in three separate families: the 144 primary comparisons, 364 comparisons in the fourteen-model population, and 36 within-checkpoint effort comparisons. Other intervals are pointwise. These intervals describe variation across questions for the evaluated models and recorded attempts (Appendix H.2).

5 Results

We first examine answer outcomes and the cost of correct responses on shared questions. We then test whether relative costs change with workload, and how reasoning effort changes these patterns.

5.1 Finding 1: Generation-cost gaps persist among successful answers

When one model generates more tokens than another, the difference can lie in unsuccessful responses, successful answers, or both. These cases distinguish two targets for efficiency evaluation: reducing expenditure on failures and reducing the cost of correct answers. We first measure the tokens associated with each answer outcome, then compare correct-response lengths on questions that both models solve.

Rare outcomes, large expenditure. Across the reasoning effort experiment, unavailable answers account for 1.29% of responses but 10.79% of generated tokens; correct and incorrect available answers account for 56.6% and 32.6%. This expenditure is concentrated. Of the 210 setting–task evaluations, 163 return available answers on more than 99% of attempts, yet 32 spend more than 10% of their tokens on unavailable answers (Fig. 1a). Most of these tokens come from responses stopped at the output cap, which alone consume 9.67% of all generated tokens (Appendix G.1).

Successful answers remain costly. To test whether failed outcomes fully explain cross-model cost differences, we compare correct-response lengths on questions that both models solve at least once. Across the 360 model-pair comparisons among the nine checkpoints (36 pairs on ten benchmarks),

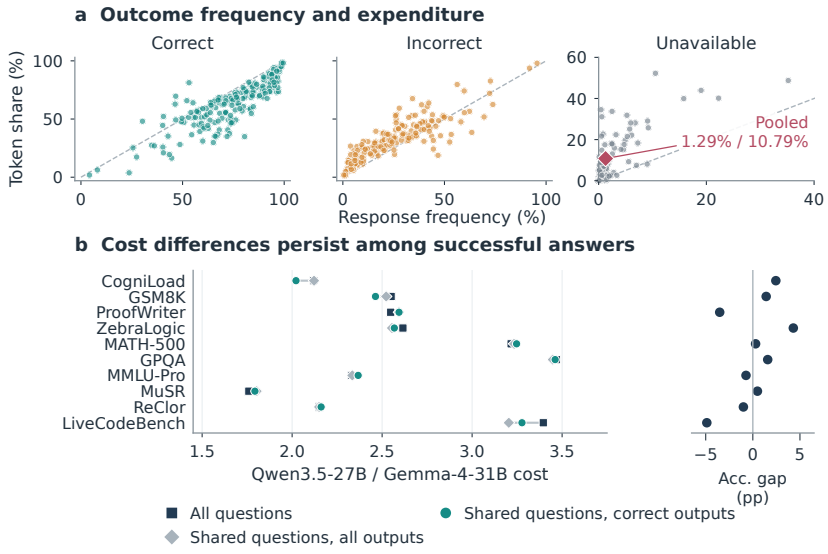

Figure 1: **Outcome frequency and token expenditure reveal persistent successful-answer cost gaps.** (a) Each circle represents one of 210 setting–task evaluations. Points above the diagonal indicate outcomes that consume a larger share of tokens than their share of responses. The diamond pools the reasoning effort experiment: unavailable answers account for 1.29% of responses and 10.79% of tokens. (b) Qwen3.5-27B/Gemma-4-31B mean-cost ratios using all outputs on all questions, all outputs on shared successful questions, and only correct outputs on those same questions. Shared successful questions have at least one correct response from each model; questions receive equal weight. The right column gives the full-benchmark accuracy difference, Qwen minus Gemma.

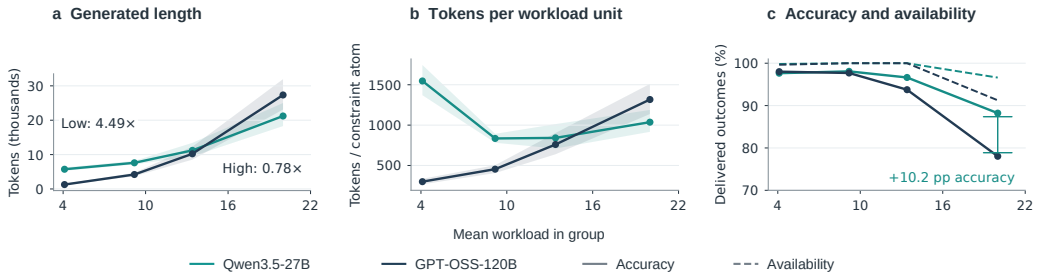

Figure 2: **Similar average lengths conceal a reversal in relative generation cost.** ZebraLogic profiles for Qwen3.5-27B (reasoning on) and GPT-OSS-120B (high effort). Panels show (a) mean generated length, (b) mean tokens per constraint atom, and (c) accuracy (solid) and answer availability (dashed). Qwen's mean generated length is 4.49 times GPT-OSS's in the lowest-workload group but 0.78 times in the highest-workload group, where Qwen is also 10.2 percentage points more accurate. The ratios in (a) compare group mean lengths. Each group contains the same questions for each model and preserves workload ties. Shading in (a–b) gives pointwise 95% intervals.

the median larger/smaller cost ratio is 1.51 on all questions and 1.54 among correct outputs on shared successful questions (Appendix G.1). Generation-cost differences therefore persist among correct answers to shared questions. For example, Qwen3.5-27B generates more tokens than Gemma-4-31B on all ten benchmarks, although its accuracy is higher on six tasks and lower on four. Among correct outputs on shared successful questions, Qwen's cost remains 1.79–3.46 times Gemma's (Fig. 1b).

5.2 Finding 2: Relative generation costs change with workload

Similar benchmark-average response lengths can conceal a reversal in which model is cheaper. On ZebraLogic, Qwen3.5-27B and GPT-OSS-120B average 11,560 and 10,968 generated tokens per

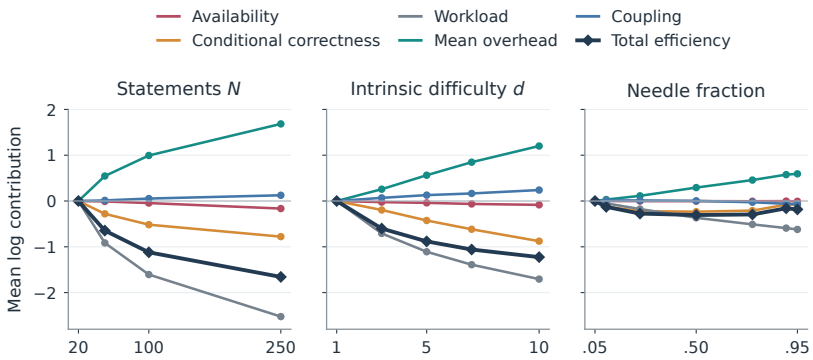

Figure 3: **Separating workload growth, normalized generation, and answer outcomes on CogniLoad.** Each comparison varies one generator variable (N, d, or needle fraction) while equally weighting combinations of the other two. Curves show mean signed log contributions across 31 models, relative to the lowest factor level, and sum to the mean log-efficiency change. The mean-overhead term is $-\Delta \log \overline{\mathrm{VO}}$, so positive values indicate fewer tokens per workload unit. For increasing statement count, this positive contribution accompanies a larger workload, deteriorating answer outcomes, and a net efficiency decline.

question, respectively. Yet Qwen generates 4.49 times as many tokens as GPT-OSS in the lowest-workload group and only 0.78 times as many in the highest-workload group (Fig. 2a). These ratios compare group mean lengths. In the highest-workload group, Qwen is also 10.2 percentage points more accurate and has higher answer availability (Fig. 2c). Thus, a comparison based on average length misses a workload group in which Qwen has an advantage in both cost and answer quality.

Evidence across model pairs. We compare all 36 pairs of the nine reasoning-enabled models on each of four annotated benchmarks, including all recorded attempts in each question's mean response length. From the within-question cost ratios, we compute D_{mn} (Eq. (5)) and summarize the magnitude of change by $\exp(|D_{mn}|)$. A value of one means equal geometric-mean cost ratios in the low- and high-workload groups; two means that the ratio is twice as large in one group as in the other. Such changes need not reverse which model is cheaper.

Simultaneous 95% intervals for D_{mn}, adjusted across all 144 comparisons, exclude zero in 82 cases. ZebraLogic has the largest median $\exp(|D_{mn}|)$, 2.43, with 34 of 36 comparisons supported. The corresponding medians and supported counts are 1.51 (25/36) for CogniLoad, 1.26 (22/36) for ProofWriter, and 1.10 (1/36) for GSM8K. Workload-dependent cost differences thus extend beyond the illustrative model pair, with substantial variation in magnitude and statistical support across benchmarks. Complete comparisons and sensitivity analyses appear in Appendix G.2. Appendix Fig. 5 shows the corresponding generated-length, tokens-per-workload-unit, accuracy, and availability profiles.

Controlled task variation through CogniLoad. Native workloads change several task properties together, so they do not show which property drives the workload response. A reduction in tokens per workload unit can occur even when responses become longer and less accurate. We examine this distinction using CogniLoad, whose generator separately varies statement count, intrinsic difficulty, and distractor content. When varying one factor, we give equal weight to combinations of the other two, holding their distribution fixed.

Increasing statement count from $N = 20$ to $N = 250$ raises mean declared workload 12.48-fold. Across settings, the median increase in generated length is only 2.24-fold, while median accuracy falls by 51.1 percentage points. Models therefore generate more text, but fewer tokens per workload unit, as answer quality deteriorates.

The decomposition in Fig. 3 connects these changes to efficiency. Falling mean overhead contributes positively to the log-efficiency change, but increasing workload and deteriorating answer outcomes

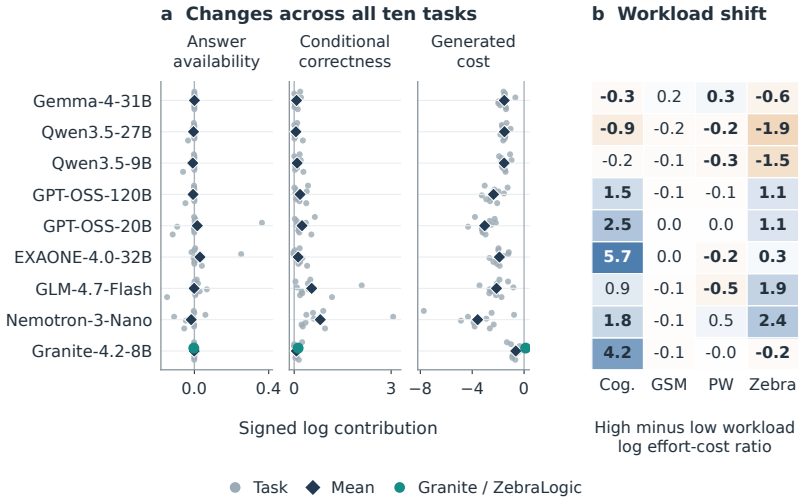

Figure 4: **Reasoning effort changes answer outcomes and the workload dependence of cost.** (a) Signed log contributions for all 90 within-checkpoint comparisons. Points represent benchmarks and diamonds average the ten benchmarks within each checkpoint. Positive contributions favor higher-effort efficiency. (b) Relative-cost contrast D_{mn} (Eq. (5)) of the higher- versus lower-effort setting: the log ratio of the geometric-mean effort multipliers at high and low workload. Positive values indicate a larger proportional cost increase at high workload; negative values indicate a larger increase at low workload. Bold entries have simultaneous 95% intervals excluding zero across the 36 comparisons. Section 5.3 connects the ZebraLogic responses to mean overhead and coupling.

outweigh that contribution. Because this comparison changes the workload distribution, the accounting includes $-\Delta \log \mathbb{E}[W]$ alongside the four terms in Eq. (4). Lower normalized generation therefore does not, by itself, establish an efficiency improvement.

Extending the workload scale. To extend the analysis to benchmarks without native annotations, we construct workload from successful reference-model outputs. We assess whether this solver-derived scale preserves differences between models in how generated length changes from low to high workload, evaluating both scales on the same covered questions.

The correlations between model-specific changes in mean log length under the native and solver-derived scales are 0.976 on CogniLoad, 0.920 on ProofWriter, and 1.000 on ZebraLogic, compared with 0.583 on GSM8K. Agreement exceeds shuffled-workload controls on the first three tasks and broadly survives whole-family reference removal. These correlations describe agreement in relative patterns across models. Coverage, differences in effect magnitude, and results on six additional benchmarks are reported in Appendix G.4.

5.3 Finding 3: Reasoning effort changes answer outcomes and the response to workload

We compare reasoning effort settings within each checkpoint, holding questions, sampling parameters, and output caps fixed, to assess what additional generation buys and how its cost varies with workload. This is of interest because it represents the main control for users of LLMs to trade cost for accuracy.

Effort trades efficiency for accuracy. Across all 90 within-checkpoint effort comparisons, accuracy rises in 87 but efficiency falls in 89. Gains in conditional correctness generally accompany higher generated cost, while availability changes concentrate in particular model–task settings (Fig. 4a). In 88 of the 89 decreases, the cost increase is so large that even perfect high-effort accuracy would not restore low-effort efficiency (Table 9). Granite on ZebraLogic provides the exception: higher effort raises accuracy by 9.3 percentage points while cutting cost by 11.5%, improving efficiency 1.27-fold.

Answer availability can also reverse the gain from higher conditional correctness. In the separate math/code experiment with a fixed 16,384-token output cap, increasing GPT-OSS-20B's reasoning effort raises LiveCodeBench correctness among available answers from 74.1% to 93.8%, but lowers availability from 97.9% to 72.8%. Overall accuracy falls from 72.5% to 68.3%, while mean generated length increases 7.3-fold. Competition mathematics shows a different pattern: large accuracy gains accompany still larger proportional cost increases. Appendix G.1 reports the scoring-sensitivity analyses, and Appendix H.1 distinguishes answer validity from stopping.

Effort reshapes the response to workload. Increasing reasoning effort can change the workload profile of generation, not just its average cost. We measure this with the relative-cost contrast D_{mn} of Eq. (5), taking m as the higher-effort and n as the lower-effort setting of the same checkpoint. Positive D_{mn} means that higher effort adds proportionally more tokens to high-workload questions. On ZebraLogic, $\exp(D_{mn})$ is 0.14 for Qwen3.5-27B and 11.2 for Nemotron. The proportional cost increase is therefore larger on low-workload questions for Qwen and on high-workload questions for Nemotron. This difference is part of a broader pattern: simultaneous 95% intervals for D_{mn} exclude zero in 21 of 36 within-checkpoint comparisons (Fig. 4b).

The full decomposition quantifies how changes in mean overhead and coupling combine in pooled cost (Table 10). For Qwen, switching reasoning on multiplies mean tokens per constraint atom by approximately 3.8, but reduces coupling to 0.74 of its previous value. Mean generated cost therefore increases by about $3.8 \times 0.74 \approx 2.8$. For Nemotron, the corresponding factors are approximately 47 and 1.62, giving a 76-fold increase in mean cost. Thus, coupling partly offsets the increase in mean normalized generation for Qwen but reinforces it for Nemotron. Their negative log changes are the two cost contributions in Eq. (4): the workload contrast identifies differences between endpoint groups, while the decomposition quantifies how workload association contributes to aggregate efficiency.

6 Limitations

Operating point. We measure generated-token efficiency for specific models, settings, and generation limits. Availability depends on the scoring rule and context limit (Appendices F and E). Tokens do not translate directly into comparable latency, energy, or monetary cost. *Workload.* Native scales describe benchmark-specific task structure, not a universal difficulty scale. Solver-derived workload additionally depends on its reference pool and coverage. *Interpretation.* The decomposition accounts for observed outcomes and costs, not which reasoning steps are necessary or which intervention would remove a cost gap. Comparisons among correct outputs describe the observed shared-success questions, rather than equal underlying model capability.

7 Conclusion

Reasoning-model cost is neither solely a consequence of failed attempts nor adequately described by a fixed verbosity multiplier. Cost gaps persist among correct answers to shared questions, while relative costs and the effects of reasoning effort change across workloads. These two findings are compatible: a model can have consistently costly successful answers yet a cost disadvantage that varies substantially with task structure. Our outcome and workload layers connect these patterns to aggregate efficiency, separating answer delivery and correctness from mean normalized generation and its association with workload.

For model development, this changes what an efficiency improvement should demonstrate. Beyond a better average score, evaluation should establish whether a change reduces unsuccessful expenditure, makes correct answers cheaper on shared questions, or shifts cost and accuracy between workload groups. The same comparisons can assess training and inference changes without inspecting reasoning traces. Extending workload annotations to more benchmarks would make these evaluations possible across a broader range of tasks. Progress in efficient reasoning should be assessed by which answers improve and where generation costs change, rather than by shorter outputs alone.

AI USE STATEMENT

Each component of this study is human designed. The idea and strategy for this research was fully devised by humans. The structure of the paper was designed by humans and the first draft written by hand. Subsequent iterations on the paper used LLM tools to polish and improve the wording of paragraphs. The final version was only edited by humans.

Generative AI tools (OpenAI Codex and Anthropic Claude Code) were used during writing to help polish the text, spellcheck, consistency check, and verify the mathematics. In addition they were used to implement the cluster jobs that generated the inference dataset on the LUMI supercomputer, the code that generated the charts from results, the implementation of the statistical analyses in the paper and appendix, and to collect the reproducibility package for reviewers. The code which calculates the key figures that the text and charts in the paper are based on was checked by humans.

We take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We provide an anonymous reviewer package containing the scored response-level measurements, evaluation protocols, and analysis code, with instructions to reproduce the reported statistics, tables, and figures without additional model inference. The package includes expected outputs and validation checks. We will publicly release the full dataset of model responses and a dedicated python package with the camera-ready version of the paper.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. In *Conference on Language Modeling (COLM)*, 2025. arXiv:2503.04697.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025. doi: 10.52202/085713-2032. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/579b5b84311e122584dedab1d3b7613f-Abstract-Conference.html. arXiv:2502.04463.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0679. URL https://proceedings.nips.cc/paper_files/paper/2025/hash/1d27c01ebd3e3aebe226b44fc970d803-Abstract-Datasets_and_Benchmarks_Track.html.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=? on the overthinking of long reasoning models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 9487–9499, 2025. URL https://proceedings.mlr.press/v267/chen25bx.html.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Zheng Du, Hao Kang, Song Han, Tushar Krishna, and Ligeng Zhu. OckBench: Measuring the efficiency of LLM reasoning. *arXiv preprint arXiv:2511.05722*, 2025. URL https://arxiv.org/abs/2511.05722.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/94074dd5a072d28ff75a76dabed43767-Abstract-Conference.html.

Daniel Kaiser, Arnoldo Frigessi, Ali Ramezani-Kebrya, and Benjamin Ricaud. CogniLoad: A synthetic natural language reasoning benchmark with tunable length, intrinsic difficulty, and distractor density. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/10b7e27c8eb9571fbbd2ae6a9f8c3855-Abstract-Conference.html.

Jinu Lee and Julia Hockenmaier. Evaluating step-by-step reasoning traces: A survey. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 1789–1814, 2025. doi: 10.18653/v1/2025.findings-emnlp.94. URL https://aclanthology.org/2025.findings-emnlp.94/.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. ZebraLogic: On the scaling limits of LLMs for logical reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 37889–37905, 2025. URL https://proceedings.mlr.press/v267/lin25i.html.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. CoT-Valve: Length-compressible chain-of-thought tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6025–6035, 2025. doi: 10.18653/v1/2025.acl-long.300. URL https://aclanthology.org/2025.acl-long.300/.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, 2025. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/.

Prasanna Parthasarathi, Mathieu Reymond, Boxing Chen, Yufei Cui, and Sarath Chandar. GRPO-λ: Credit assignment improves LLM reasoning. *arXiv preprint arXiv:2510.00194*, 2025. URL https://arxiv.org/abs/2510.00194.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *Conference on Language Modeling*, 2024. URL https://colmweb.org/2024/AcceptedPapers.html.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b623663fd9b874366f3ce019fdfdd44-Abstract-Conference.html.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. MuSR: Testing the Limits of Chain-of-Thought with Multistep Soft Reasoning. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.16049.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*, 2025. URL https://arxiv.org/abs/2503.16419.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, 2021. doi: 10.18653/v1/2021.findings-acl.317. URL https://aclanthology.org/2021.findings-acl.317/.

Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. Reasoning in token economies: Budget-aware evaluation of LLM reasoning strategies. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 19916–19939, 2024a. doi: 10.18653/v1/2024.emnlp-main.1112. URL https://aclanthology.org/2024.emnlp-main.1112/.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024b. URL https://arxiv.org/abs/2406.01574.

Zihao Wei, Liang Pang, Jiahao Liu, Jingcheng Deng, Shicheng Xu, Zenghao Duan, Jingang Wang, Fei Sun, Xunliang Cai, Huawei Shen, and Xueqi Cheng. Stop spinning wheels: Mitigating LLM overthinking via mining patterns for early reasoning exit. *arXiv preprint arXiv:2508.17627v1*, 2025. URL https://arxiv.org/abs/2508.17627v1.

Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. Towards thinking-optimal scaling of test-time compute for LLM reasoning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1452. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3e22bea3b170f4c2aebb9c48d98ae64d-Abstract-Conference.html.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. ReClor: A Reading Comprehension Dataset Requiring Logical Reasoning. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/2002.04326.

A ACCOUNTING IDENTITIES AND ESTIMATORS

This appendix derives the accounting identities, defines the workload scales, and documents the evaluation and reproduction protocols.

A.1 EFFICIENCY AND ANSWER ACCEPTANCE

For a response to instance I, let $S \in \{0, 1\}$ indicate correctness and let $T > 0$ be its recorded generated-token count. When the scorer exposes answer acceptance, let $A \in \{0, 1\}$ indicate that an answer was accepted for scoring. An accepted answer may be correct or incorrect. Every correct answer is accepted, so $S \leq A$. The cost denominator includes all recorded generation, including rejected and incorrect answers.

For n equally weighted responses, pooled accuracy-per-token is

$$\widehat{E}_0 = 1000 \frac{\sum_{i=1}^{n} S_i}{\sum_{i=1}^{n} T_i}. \tag{6}$$

This ratio of sums measures successful answers per total generated cost. With a fixed number of attempts per question, response-level averages give each question equal weight. For unequal repeat counts, declared question and attempt weights define the pooled estimate.

Define $r_a = \Pr(A = 1)$ and $r_c = \Pr(S = 1 \mid A = 1)$. Since a correct answer is accepted,

$$\Pr(S = 1) = r_a r_c, \qquad E_0 = \frac{1000 r_a r_c}{\mathbb{E}[T]}. \tag{7}$$

The empirical rates are $\widehat{r}_a = \sum_i A_i / n$ and $\widehat{r}_c = \sum_i S_i / \sum_i A_i$.

For a model and reference evaluated on the same questions with the same weights, dividing their efficiencies gives

$$\frac{E_0}{E_{0,\mathrm{ref}}} = \frac{r_a}{r_{a,\mathrm{ref}}} \frac{r_c}{r_{c,\mathrm{ref}}} \frac{\mathbb{E}_{\mathrm{ref}}[T]}{\mathbb{E}[T]}.$$

The factor of 1,000 cancels. Taking logarithms and defining $\Delta \log x = \log(x / x_{\mathrm{ref}})$ for positive factors yields

$$\Delta \log E_0 = \Delta \log r_a + \Delta \log r_c - \Delta \log \mathbb{E}[T]. \tag{8}$$

Logs turn a product of ratios into additive contributions. A negative signed contribution reduces efficiency relative to the reference, while a positive one increases it. The contributions describe observed outcomes and costs and may be statistically dependent.

If no answer is accepted, r_c is undefined. If no answer is correct, $E_0 = 0$ and its finite log contrast is undefined. These cases retain their raw rates and token costs, without pseudocounts. Missing acceptance metadata is preserved as missing. Correctness and length still define E_0.

Acceptance and stopping. Answer acceptance records whether a response can be scored, while the stopping reason records how generation ended. In the model comparison, the scorer checks whether the response reaches the context limit and whether a final answer can be extracted. The records do not separately identify why generation stopped. Reasoning effort and math/code records distinguish final-answer validity from stopping. Both accepted length-stopped answers and rejected normally stopped answers occur, as shown in Table 22 for the reasoning effort experiment.

A.2 WORKLOAD-NORMALIZED FACTORIZATION

Let $W(I) > 0$ be a declared workload shared by the models on each compared instance, and define verbalization overhead $\mathrm{VO} = T/W$. Since $T = W\mathrm{VO}$ pointwise,

$$\mathbb{E}[T] = \mathbb{E}[W]\mathbb{E}[\mathrm{VO}] + \mathrm{Cov}(W, \mathrm{VO}) = \mathbb{E}[W]\overline{\mathrm{VO}}\kappa, \tag{9}$$

where

$$\overline{\mathrm{VO}} = \mathbb{E}[T/W], \qquad \kappa = \frac{\mathbb{E}[T]}{\mathbb{E}[W]\mathbb{E}[T/W]}. \tag{10}$$

On the same instances with the same weights, $\mathbb{E}[W]$ cancels between models, giving

$$\Delta \log E_0 = \Delta \log r_a + \Delta \log r_c - \Delta \log \overline{\mathrm{VO}} - \Delta \log \kappa. \tag{11}$$

When acceptance is unavailable, the equally exact version uses $\Delta \log \Pr(S = 1)$ in place of the first two terms.

What mean overhead and coupling distinguish. Mean overhead averages tokens per workload unit using the declared question weights. Pooled tokens per workload unit, $\mathbb{E}[T]/\mathbb{E}[W]$, instead gives additional weight to questions with larger workload. Consequently,

$$\kappa = \frac{\mathbb{E}[W\,\mathrm{VO}]/\mathbb{E}[W]}{\mathbb{E}[\mathrm{VO}]}$$

is the workload-weighted mean overhead divided by the ordinary mean overhead. It exceeds one when larger-workload questions have higher per-unit costs in the covariance sense, and is below one when the association is negative.

For two models or settings evaluated on the same questions, weights, and workload assignments,

$$\frac{\mathbb{E}[T_m]}{\mathbb{E}[T_n]} = \frac{\overline{\mathrm{VO}}_m}{\overline{\mathrm{VO}}_n} \frac{\kappa_m}{\kappa_n}.$$

Thus, a difference in mean normalized generation can be reinforced or partly offset by its association with workload. The two factors account for pooled cost; neither alone determines the low–high workload contrast in Eq. (5).

Changing units to $W' = aW$ for fixed $a > 0$ divides $\overline{\mathrm{VO}}$ by a and leaves κ and the log contrasts unchanged. We compare workload-normalized quantities within tasks, where their units agree. Averaging task-local log contrasts gives the log of a geometric mean of task-level efficiency ratios. Pooling all responses defines a separate estimand.

Alternative workload definitions can change the separate overhead and coupling contributions while leaving their sum, $-\Delta \log \mathbb{E}[T]$, unchanged on matched questions. The endpoint contrast D_{mn} instead depends on workload only through the membership of the low- and high-workload groups.

B Native workload definitions

Workload provides an external, benchmark-specific scale for comparing generated length. The units below represent problem structure or reference-solution structure. They are abstractions of the task rather than measurements of model-internal computation.

CogniLoad. Each instance specifies an initial state, sequential conditional updates, and a query about a person of interest (PoI). The workload is

$$W_{\mathrm{poi}}(I) = \sum_{t=1}^{N} k_t + \sum_{t \in \mathcal{N}(I)} m_t, \tag{12}$$

where k_t counts condition clauses, m_t counts update clauses, and $\mathcal{N}(I)$ contains statements affecting the queried person. This oracle abstraction checks all conditions and tracks updates for the queried person. The controlled variables N, intrinsic difficulty d, and needle fraction η retain their generator meanings. Here needle fraction is the proportion of relevant updates.

GSM8K. The native scale is the positive count of steps in the supplied reference solution. It measures that solution's structure, which may include more steps than another valid derivation.

ProofWriter. The stored workload is $W_{\mathrm{depth}} = 1 +$ proof depth. The additive one gives depth-zero instances positive workload. In the balanced model-comparison panel, depths zero through five give workload values one through six.

ZebraLogic. The native scale counts parsed structural clue atoms, representing the stated constraint structure. Each comparison uses the same recorded annotation for every model.

Native workloads are available for these four task types. Question membership and workload equality are checked within each experiment before cancelling $\mathbb{E}[W]$. For tasks with successful reference outputs, the solver-derived scale below provides an additional normalization.

C Solver-derived workload and coverage

Successful reference outputs provide an operational workload scale. We rank a declared reference pool by benchmark-level pooled efficiency. On each instance, the highest-ranked reference configuration that answers correctly supplies its successful output length as $W_{\mathrm{solv}}(I)$. Thus benchmark-level rank determines the selected solver, whose response length defines the workload.

The reference population, scoring rule, and generation protocol define this scale. Workload-profile agreement, constant and permuted controls, and whole-family removal assess this operational choice (Section G.4). Mean-overhead ranking checks supply a complementary sensitivity analysis.

Unsolved instances. If no eligible reference succeeds, W_{solv} is undefined. Those instances remain in full-population accuracy and efficiency reports. Solver-workload analyses report coverage and recompute every term of the identity on the covered subset, using identical questions and weights for every evaluated configuration.

Repeated attempts and held-out responses. Reference ranking, reference success, and per-question workload use designated reference attempts. The resulting workload is frozen for evaluation. The single-response model-comparison experiment uses the same response for both stages. Reasoning effort reserves attempt zero for reference construction and evaluates the remaining attempts. Competition math reserves its first recorded attempt and evaluates the remaining three. These repeated-response experiments separate reference and evaluation responses while retaining the same checkpoint pool.

Sensitivity checks. The removal audit deletes each reference configuration in turn and reports the weakest agreement with the baseline ranking across these deletions. Each deletion defines a common workload for the full evaluation ranking. A second perturbation removes the five highest-ranked references together. For each perturbation, baseline and perturbed rankings use their shared covered questions. The audit reports the retained question count alongside rank agreement.

D Selection and generated-cost audits

D.1 Comparing accepted answers on the same instances

Each model's r_c conditions on its own accepted subset. To compare correctness on shared questions, let J be the set on which both answers are accepted in a single-response model–reference comparison. The difference in total correct answers partitions exactly as

$$\begin{aligned}\sum_i S_{i,m} - \sum_i S_{i,\mathrm{ref}} = \sum_{i\in J}(S_{i,m} - S_{i,\mathrm{ref}}) \\ + \sum_{i:A_{i,m}=1,A_{i,\mathrm{ref}}=0} S_{i,m} - \sum_{i:A_{i,m}=0,A_{i,\mathrm{ref}}=1} S_{i,\mathrm{ref}}.\end{aligned} \tag{13}$$

The first term compares both systems on identical questions, while the other terms account for correct answers outside that intersection. We report the two conditional correctness rates on J together with the full-population counts.

For repeated-response extensions, a matched comparison requires either a prespecified attempt pairing or question-level conditional rates on questions with accepted outputs from both configurations. The choice defines the selected population. Attempt indices alone imply no trajectory pairing. Such comparisons supplement full-population accuracy with an analysis of the observed accepted answers.

Experiment	Settings	Tasks	Responses	Attempts/question
Model comparison	31 / 14	4	115,541	1
Reasoning effort	21	10	597,744	8 / 16
Math/code	4	2	65,536	64
Competition math	28	2	6,720	4
Total			785,541	

Table 2: Response inventory. Model comparison has 31 settings on CogniLoad and 14 on each other task. Reasoning effort uses eight attempts on six tasks and sixteen on four. Math/code contains two execution blocks of 32 attempts. Competition math uses the MathArena configurations with complete token costs. Response counts include repeated attempts and overlapping checkpoint populations.

D.2 Separating rejected-answer expenditure

A rejected response both contributes no successful answer and incurs generated cost. To collect this frequency-and-cost burden into one term, define

$$\mu_A = \mathbb{E}[T \mid A = 1], \qquad \mu_R = \mathbb{E}[T \mid A = 0], \qquad B_R = 1 + \frac{1 - r_a}{r_a}\frac{\mu_R}{\mu_A}. \tag{14}$$

Then, for $r_a > 0$,

$$\mathbb{E}[T] = r_a\mu_A + (1 - r_a)\mu_R, \qquad E_0 = \frac{1000 r_c}{\mu_A B_R}. \tag{15}$$

When every answer is accepted, the rejected-cost contribution is zero and $B_R = 1$, without evaluating a mean over the empty rejected subset. This factorization separates mean accepted-answer cost μ_A from the additional expenditure on rejected answers, B_R.

An additional direct audit partitions token spending into correct accepted answers, incorrect accepted answers, and rejected answers:

$$\pi_{\text{correct}} = \frac{\mathbb{E}[TS]}{\mathbb{E}[T]}, \quad \pi_{\text{wrong}} = \frac{\mathbb{E}[T(A - S)]}{\mathbb{E}[T]}, \quad \pi_{\text{rejected}} = \frac{\mathbb{E}[T(1 - A)]}{\mathbb{E}[T]}. \tag{16}$$

These nonnegative shares sum to one, assigning every recorded output token to its response's scored outcome.

E Budgets, censoring, and physical stopping

The reasoning effort experiment caps generated output at 81,920 tokens, and the math/code experiment at 16,384. The model-comparison experiment includes several context allowances. Output caps apply to generated tokens, whereas context limits apply to the combined input and output. Each comparison describes its specified operating points.

Retrospective censoring tests whether a recorded trajectory fits a shorter hypothetical allowance. A fresh generation under a shorter budget is a separate experiment, allowing the trajectory and stopping policy to change. A smaller combined context limit may also exclude longer prompts, changing the eligible question population. Common question identities are therefore required for a comparison that holds question composition fixed.

The two coding experiments differ in questions, settings, and generation protocol as well as their caps. They provide complementary evaluations of those operating points. Competition math uses the public MathArena responses at their recorded full-response costs.

F Evaluation sets, generation, and scoring

The study comprises four experiments: model comparison, reasoning effort, math/code, and competition math. Each specifies its questions, generation settings, and scoring protocol. A setting identifies a checkpoint and its generation control, allowing several operating points of one model.

Checkpoint	Reasoning settings	Temp.	Top-p	Top-k	Presence
GPT-OSS-20B	low / medium / high	1.0	1.00	off	0.0
GPT-OSS-120B	low / medium / high	1.0	1.00	off	0.0
Qwen3.5-9B	off / on	1.0	0.95	20	1.5
Qwen3.5-27B	off / on	1.0	0.95	20	1.5
GLM-4.7-Flash	off / on	1.0	0.95	off	0.0
EXAONE-4.0-32B	off / on	0.6	0.95	off	0.0
Nemotron-3-Nano-30B-A3B	off / on	1.0	0.95	off	0.0
Granite-4.2-8B	off / brief / on	1.0	0.95	off	0.0
Gemma-4-31B	off / on	1.0	0.95	64	0.0

Table 3: Sampling settings held fixed within each checkpoint in the reasoning effort experiment. Off in the Top-k column denotes disabled top-k filtering (stored as -1), which is separate from the thinking control. Presence denotes the presence penalty.

F.1 Model comparison

The CogniLoad set contains 1,325 questions evaluated by 31 settings, including six GPT-OSS settings. The other shared tasks contain 1,319 GSM8K questions, 3,000 ProofWriter questions, and 1,000 ZebraLogic questions for fourteen models. Each decomposition uses identical questions and workload assignments within its stated comparison.

The ZebraLogic scorer handles house-index aliases in the 1,000-question panel. The generation audit for the GPT-OSS factorial runs records context settings of 32,768, 43,008, and 53,248 tokens. These settings are documented in the protocol metadata. Missing per-response cap fields remain missing. The acceptance and stopping fields follow the definitions in Section A.

F.2 Reasoning effort

Nine checkpoints supply 21 reasoning settings on ten tasks. GSM8K, MATH-500, ProofWriter, ZebraLogic, and CogniLoad each have 256 questions, and GPQA has 198. These six tasks have eight attempts per setting and question. MMLU-Pro has 336 questions, MuSR 192, and ReClor and LiveCodeBench 256 each. These four tasks have sixteen attempts. This gives 2,518 question identities and 597,744 responses.

GPT-OSS uses its built-in reasoning effort control, while the other checkpoints use their thinking controls. Sampling is fixed within each checkpoint across settings. All runs have an 81,920 generated-token cap and a 98,304 context limit with a 16,384 prompt allowance. Native end-of-sequence behavior is retained. EXAONE and Nemotron use the qualified vLLM 0.17.1 NVIDIA image. The remaining completed runs use the qualified vLLM 0.25.1 ARM64 image.

For MMLU-Pro, MuSR, ReClor, and LiveCodeBench, questions were selected in a fixed hash order before observing responses. MMLU-Pro is balanced across fourteen subjects. MuSR uses distinct stories and ReClor distinct passages. LiveCodeBench uses contest problems from the pinned October 2024–April 2025 additions, excluding the question identities in the math/code experiment. The records retain ProofWriter parent groups and LiveCodeBench contest groups for clustered uncertainty calculations.

The category scorer retains outputs accepted by the strict extractor. For strict-unaccepted GPQA/ProofWriter outputs, it extracts a terminal boxed category or terminal category label without consulting gold, then scores the extracted candidate. This fallback produces 4,378 extracted candidates.

F.3 Math/code

The math/code experiment contains GPT-OSS-20B at low/high effort and Qwen3.5-9B with thinking off/on, evaluated on 128 MATH-500 and 128 LiveCodeBench questions. Two execution blocks each provide 32 responses per question and setting, yielding 65,536 responses. The cap is 16,384 generated tokens within a 32,768-token context. These evaluations use vLLM 0.25.1 on ARM64

and the same checkpoint revisions and sampling parameters for these two models as in Table 3. The coding questions are disjoint from those in the reasoning effort experiment.

The coding protocol extracts the final channel's last fenced code block and uses the pinned official LiveCodeBench checker, commit `28fef95ea8c9f7a547c8329f2cd3d32b92c1fa24`. It runs all pinned public and private tests, allowing the checker to stop after failure, with the dataset's declared entry point. The recorded sandbox uses Python 3.11, one CPU, 2 GiB memory, and a six-second per-test limit. Program failures and grader infrastructure errors remain distinct. Generation produces a single response under the model's native template, without iterative tool feedback.

The math scorer extracts the last balanced boxed answer, otherwise the last explicit answer line. An incomplete final box fails extraction. Answer equivalence is checked with Math-Verify 0.8.0, with strict verification and separate five-second parse and verify limits. Scorer errors retain missing correctness until resolved. Analysis requires a complete set of scores.

F.4 Competition math

The competition-math experiment analyzes public MathArena responses for AIME 2025 and HMMT February 2025 (Balunović et al., 2025), each with thirty problems and four sampled attempts per configuration. The 28 configurations with complete per-attempt token costs on both tasks supply 6,720 responses. These evaluations use MathArena's prompts and provider settings, which differ from the other experiments.

The available fields are correctness and token costs. Acceptance and native stop metadata are missing. Competition-math results therefore report accuracy, generated cost, efficiency, and solver normalization on the covered questions.

G Complete diagnostic analyses

This appendix describes the complete comparison populations, uncertainty estimates, and sensitivity analyses. The primary workload population comprises nine checkpoints evaluated with reasoning on (high effort for GPT-OSS). The fourteen models observed on all four native-workload tasks supply a separate corroborating population.

G.1 Outcomes, expenditure, and comparisons on shared questions

For each setting–task evaluation, we report outcome frequencies $\mathbb{E}[S]$, $\mathbb{E}[A - S]$, and $\mathbb{E}[1 - A]$, and the token-spending shares in Section D. These partitions use the same question and attempt weights as the efficiency estimate. The analysis covers 347 evaluations, of which 291 have recorded answer-availability labels. For the remaining 56 competition-math evaluations, correctness and unsuccessful-response spending remain available, while the unavailable/incorrect-eligible split is undefined. Collection-wide expenditure totals sum recorded tokens across the stated population. They do not give each task or setting equal weight.

Write $s_{m,i}$ and $a_{m,i}$ for question-level mean correctness and availability, and $c_{m,i} = s_{m,i}/a_{m,i}$ where $a_{m,i} > 0$. Three distinct comparisons are retained:

$$r_{c,m} = \frac{\sum_i s_{m,i}}{\sum_i a_{m,i}}, \qquad \widetilde{r}_{c,m} = \frac{1}{|I_m|} \sum_{i \in I_m} c_{m,i}, \qquad \widetilde{r}_{c,m|J} = \frac{1}{|J|} \sum_{i \in J} c_{m,i}, \tag{17}$$

where $I_m = \{i : a_{m,i} > 0\}$ and $J = I_m \cap I_n$. The first weights eligible questions by their availability. The second instead weights a model's eligible questions equally, and the third applies those equal weights to the shared eligible questions. For a single attempt, the first two coincide. Comparing the second with the third isolates support selection from the change in weighting. Reversal counts in Table 4 are descriptive. Overlapping pairs are not independent replications.

For cost comparisons, define the shared successful support $K = \{i : s_{m,i} > 0,\ s_{n,i} > 0\}$ and the question-level correct-output mean

$$\overline{T}^{S}_{m,i} = \frac{\sum_j T_{m,i,j} S_{m,i,j}}{\sum_j S_{m,i,j}}. \tag{18}$$

Table 4: Descriptive correctness-order reversals when moving to shared eligible questions. The first column of counts starts from pooled conditional correctness, while the second gives equal weight to questions on both the own and shared eligible supports. Ties and undefined comparisons are not counted as reversals.

Population	Pairs	Pooled to shared	Equal own to shared
Model comparison	738	16	16
Nine models	360	7	2
Math/code	12	1	0

We retain three cost comparisons for each model pair: all responses on the full question set, all responses on the shared successful set K, and correct responses on that same set K. In every comparison, response lengths are first averaged within each question and then equally across questions.

The first restriction changes only which questions are included. The second changes only which response outcomes are included, holding those questions fixed. Comparing all-response and correct-response cost ratios on K therefore tests whether the observed cost gap persists after unsuccessful responses are excluded, rather than confounding that restriction with a different question population.

A question belongs to K when each model has at least one recorded correct response. Questions retain equal weight regardless of how many successful attempts each model produces. The comparison describes the cost of observed successful answers on this shared set; full-population accuracy remains a separate measure of capability. Attempt indices are not paired between models.

All 3,606 within-experiment model/setting pairs retain their shared memberships, including pairs outside the primary nine-model population.

In the reasoning effort experiment, 163 of 210 setting–task evaluations have availability above 99%. Correct and incorrect eligible responses account for 56.6% and 32.6% of total generation. Unavailable answers account for 10.79%.

Qwen3.5-27B's full-population cost ratio to Gemma-4-31B is 1.76–3.47 across ten tasks (median 2.55), with higher accuracy on 6 tasks and lower accuracy on 4. Their shared successful sets contain 156–297 questions. Comparing all outputs on these sets with correct outputs on the same sets separates changes in question support from the restriction to successful responses. Across all 360 pairwise comparisons among the nine models, full-population and shared-success cost orderings differ in 13 comparisons.

Scoring and stopping sensitivity. The strict-label analysis replaces the GPQA and ProofWriter terminal-category fallback with retained strict extraction and correctness labels. All other tasks retain their scoring rule. It reproduces 89 efficiency decreases among 90 effort comparisons, with accuracy increases in 85 rather than 87. The accuracy-ceiling condition holds in 87 rather than 88 comparisons. For these nine models, pooled-to-shared correctness reversals change from 7 to 8, while the equal-question support reversals remain 2. Workload-cost contrasts use recorded lengths and annotations, so their values are unaffected by scoring.

Among reasoning effort responses, 3,391 unavailable length-stopped outputs consume 9.67% of all generated tokens. A further 4,319 unavailable normally stopped outputs consume 1.11%. Table 22 summarizes answer validity and recorded stopping reasons. These are expenditures associated with scored outcomes. The model-comparison and competition-math records do not identify native stopping causes.

Under strict scoring, Qwen3.5-27B's shared-success cost ratio to Gemma-4-31B remains 1.79–3.45 across the ten tasks. This persistent cost difference also appears under the retained extraction rule.

G.2 Workload response and simultaneous uncertainty

The low/high contrast in Eq. (5) uses percentile thresholds computed once from the benchmark annotations. Both endpoints include threshold ties. Table 5 gives the resulting group sizes. Profile plots use ProofWriter's discrete workloads 1–6, GSM8K's 2, 3, 4, 5, and ≥ 6 steps, and four tie-preserving groups for CogniLoad and ZebraLogic. For the latter, if q_{25}, q_{50}, q_{75} are annotation

quantiles, group 1 is $W \leq q_{25}$, group 4 is $W \geq q_{75}$, and the remaining questions are split at q_{50}. Endpoint profile groups coincide with those used by the contrast. Identical memberships apply to every model and effort setting.

Table 5: Complete workload-contrast census. Low/high group sizes include boundary ties. Supported counts use a separate simultaneous 95% family for each population. Fold shift is the median of $\exp(|D_{mn}|)$. R^2 describes the additive model for log length.

Population	Task	Low/high	Supported	Fold	R^2
Nine models	CogniLoad	64/64	25/36	1.51	0.885
Nine models	GSM8K	133/123	1/36	1.10	0.863
Nine models	ProofWriter	86/84	22/36	1.26	0.884
Nine models	ZebraLogic	69/70	34/36	2.43	0.900
Fourteen models	CogniLoad	334/332	78/91	1.87	0.381
Fourteen models	GSM8K	697/622	25/91	1.06	0.638
Fourteen models	ProofWriter	1000/1000	60/91	1.28	0.765
Fourteen models	ZebraLogic	275/291	52/91	1.17	0.890

For each task, 999 question/parent-cluster bootstrap draws share the same multiplicities across all settings and contrasts, keeping recorded attempts together. Independent tasks use independent draws. For one declared family, let $\widehat{D}_j$ be contrast j, $D^*_{b,j}$ its bootstrap value, and $\widehat{\sigma}_j$ the standard deviation of its 999 draws. We compute

$$c_{.95} = \text{quantile}_{.95}\left\{\max_j \left|\frac{D^*_{b,j} - \widehat{D}_j}{\widehat{\sigma}_j}\right|\right\}_{b=1}^{999}, \qquad \mathcal{I}_j = [\,\widehat{D}_j - c_{.95}\widehat{\sigma}_j,\ \widehat{D}_j + c_{.95}\widehat{\sigma}_j\,]. \tag{19}$$

The maximum uses a fixed bootstrap standard error for each contrast. We apply this construction separately to the 144 primary, 364 corroborating, and 36 effort/workload contrasts. It accounts for their shared questions and overlapping models within each family. Critical values are 3.446, 3.684, and 3.152. Supported counts describe this fixed comparison population rather than a binomial prevalence estimate based on independent pairs.

Relation to a fixed model-specific cost multiplier. The verbosity baseline is the balanced least-squares model

$$\log \overline{T}_{m,i} = \alpha_m + \beta_i + \epsilon_{m,i}.$$

The model effect α_m represents a fixed multiplicative cost difference, while β_i represents a shared question-specific cost profile. The fitted residual subtracts the model and question means of log length and adds the grand mean. Reported R^2 is the in-sample fraction of total variation explained over the full model–question matrix.

The workload contrast tests a specific departure from this baseline. The shared question effect cancels within each model pair, and the model effect cancels between workload groups:

$$D_{mn} = \text{mean}_{i\in H}(\widehat{\epsilon}_{m,i} - \widehat{\epsilon}_{n,i}) - \text{mean}_{i\in L}(\widehat{\epsilon}_{m,i} - \widehat{\epsilon}_{n,i}).$$

Thus, the workload contrasts measure structured differences remaining after fixed model and question effects are removed. A high overall R^2 can coexist with substantial departures within particular workload groups.

Stability across recorded attempts. Split-half sensitivity independently averages attempts 0–3 and 4–7 within each question and recomputes the complete primary contrast grid.

In the fourteen-model population, simultaneous intervals exclude zero in 215/364 workload contrasts. The primary split-half estimates have Spearman correlation 0.985 and matching signs in 136/144 comparisons. From the lowest to highest plotted workload group, median accuracy decreases across the nine models are 45.3, 7.2, and 27.1 percentage points on CogniLoad, ProofWriter, and ZebraLogic, respectively.

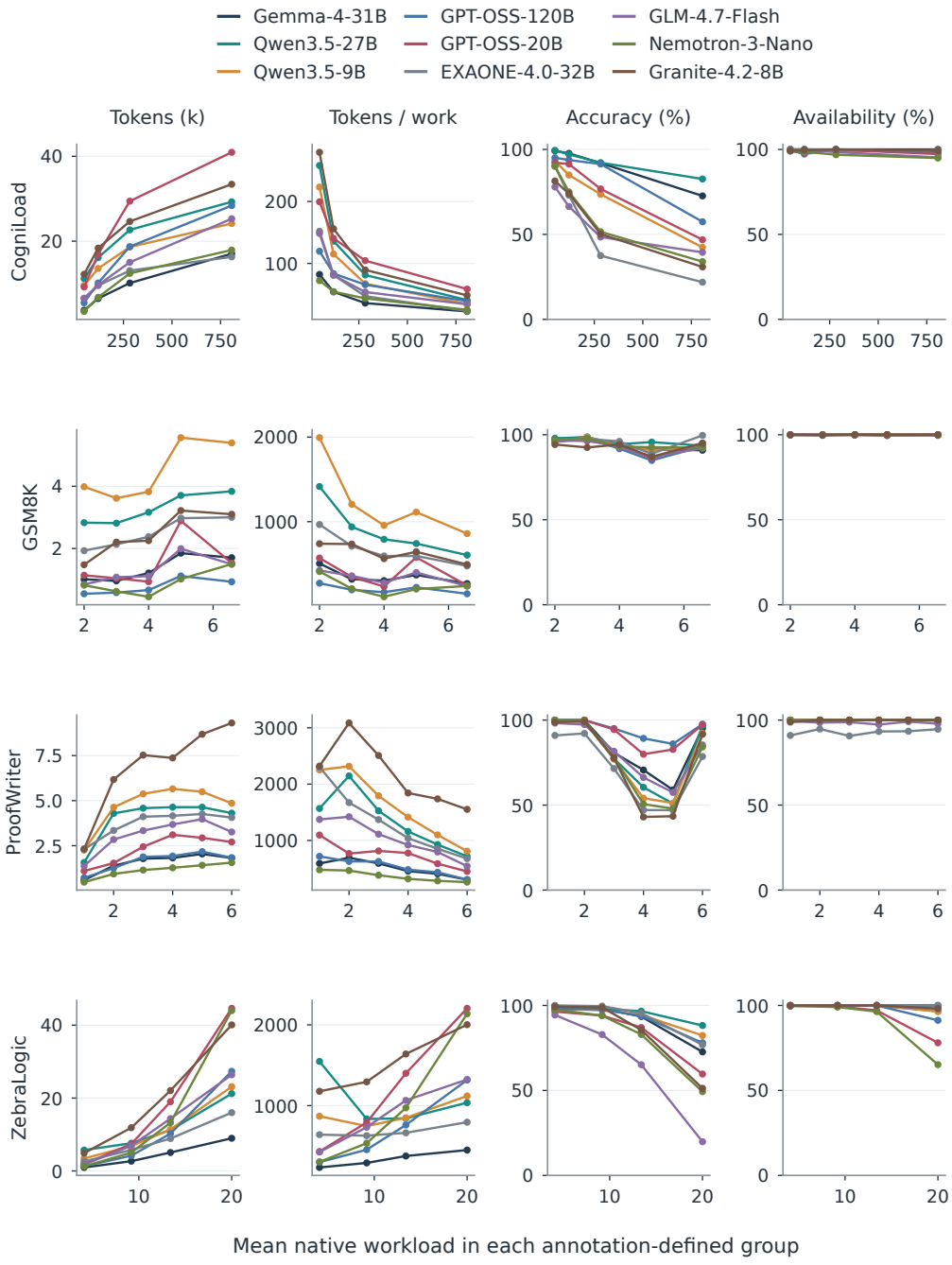

Figure 5: Complete primary workload profiles: nine models with reasoning on (high effort for GPT-OSS) on four native-workload tasks. Each row represents one task. Columns show mean generated length, mean tokens per workload unit, accuracy, and availability on identical annotation-defined groups.

Standardized CogniLoad factors. All combinations of the three generator variables are represented, with unequal observed numbers of questions per cell. For each target-variable level, we average within cells and then equally weight all combinations of the other two variables. A question in nuisance cell c receives weight $1/(Cn_c)$, where C is the number of nuisance combinations and n_c its question count. The same weights apply to all 31 settings and every moment of the decomposition.

When workload groups differ, the complete identity is

$$\Delta \log E_0 = \Delta \log r_a + \Delta \log r_c - \Delta \log \mathbb{E}[W] - \Delta \log \overline{\mathrm{VO}} - \Delta \log \kappa. \tag{20}$$

We compute the five contributions per setting and average their log values across settings. This preserves the additive identity, whereas taking a median of each component separately would not. The tables below give individual endpoint contributions using these same standardization weights.

Table 6: Individual standardized CogniLoad contributions: statements from 20 to 250. Equal nuisance-cell weights are applied within each factor level. Signed log components sum to the net change before rounding.

Setting	Avail.	Cond. correct.	Work	Overhead	Coupling	Net
EXAONE-Deep-32B	-0.01	-0.82	-2.52	+2.32	+0.20	-0.82
QwQ-32B	-0.13	-0.72	-2.52	+1.64	-0.00	-1.73
Qwen3-1.7B	+0.00	-0.36	-2.52	+2.50	-0.12	-0.51
Qwen3-30B-A3B	-0.01	-1.07	-2.52	+1.84	+0.12	-1.64
Qwen3-32B	+0.00	-1.07	-2.52	+1.69	+0.15	-1.75
Qwen3-8B	-0.02	-0.96	-2.52	+1.95	+0.02	-1.53
GLM-Z1-32B	-0.00	-1.12	-2.52	+1.82	+0.20	-1.64
MiMo-7B-RL	+0.00	-0.07	-2.52	+2.53	-0.01	-0.06
DeepSeek-R1-0528	-0.00	-1.26	-2.52	+1.75	+0.18	-1.85
R1D-Llama-70B	+0.00	-1.16	-2.52	+1.37	+0.05	-2.27
R1D-Qwen-1.5B	+0.04	-0.37	-2.52	+3.15	+0.03	+0.32
R1D-Qwen-32B	+0.00	-0.91	-2.52	+1.12	-0.08	-2.39
R1D-Qwen-7B	-0.00	-0.84	-2.52	+2.23	+0.01	-1.13
Gemini-2.5-Flash	-1.60	-0.33	-2.52	+0.98	+0.21	-3.27
Gemini-2.5-Flash-lite	-0.05	-0.62	-2.52	+1.07	+0.24	-1.88
Gemini-2.5-Pro	-1.45	-0.13	-2.52	+0.96	+0.24	-2.91
Phi-4-mini-reasoning	-0.01	-0.54	-2.52	+2.51	+0.21	-0.36
Phi-4-reasoning	-0.46	-0.82	-2.52	+1.43	+0.17	-2.19
Phi-4-reasoning-plus	-0.80	-0.66	-2.52	+1.49	+0.23	-2.27
GPT-5	-0.10	-0.18	-2.52	+1.16	+0.02	-1.63
GPT-5-mini	-0.13	-0.64	-2.52	+1.26	+0.07	-1.95
GPT-5-nano	+0.00	-1.13	-2.52	+1.83	+0.34	-1.48
GPT-OSS-120B-high	+0.00	-0.89	-2.52	+1.13	+0.22	-2.06
GPT-OSS-120B-low	+0.00	-1.06	-2.52	+2.50	+0.18	-0.90
GPT-OSS-120B-medium	+0.00	-0.86	-2.52	+1.01	+0.13	-2.24
GPT-OSS-20B-high	+0.00	-1.37	-2.52	+1.49	+0.25	-2.15
GPT-OSS-20B-low	+0.00	-0.77	-2.52	+2.25	+0.27	-0.77
GPT-OSS-20B-medium	+0.00	-1.22	-2.52	+1.20	+0.17	-2.37
o3	-0.07	-0.32	-2.52	+1.07	+0.02	-1.82
o4-mini	-0.06	-0.91	-2.52	+1.32	+0.07	-2.11
glm-4.5-air	-0.25	-0.94	-2.52	+1.61	+0.08	-2.02

G.3 Reasoning effort: components and workload response

All 90 endpoint comparisons use matched questions and fixed within-checkpoint sampling parameters: high versus low for GPT-OSS and on versus off otherwise. Table 9 summarizes the three signed log contributions over ten tasks per checkpoint. For the 36 native-task comparisons, the generated-cost contribution further separates into mean overhead and coupling. Table 10 reports each contribution and the low/high-workload contrast of the within-question log effort-cost ratio. A positive workload contrast means the high-effort cost multiplier increases with workload.

Table 7: Individual standardized CogniLoad contributions: intrinsic difficulty from 1 to 10. Equal nuisance-cell weights are applied within each factor level. Signed log components sum to the net change before rounding.

Setting	Avail.	Cond. correct.	Work	Overhead	Coupling	Net
EXAONE-Deep-32B	-0.01	-1.32	-1.71	+1.60	+0.36	-1.07
QwQ-32B	-0.11	-0.76	-1.71	+1.55	+0.05	-0.97
Qwen3-1.7B	+0.00	-1.07	-1.71	+2.59	-0.27	-0.46
Qwen3-30B-A3B	+0.00	-1.45	-1.71	+1.67	+0.16	-1.32
Qwen3-32B	+0.00	-1.07	-1.71	+1.66	+0.27	-0.84
Qwen3-8B	-0.00	-1.08	-1.71	+1.70	+0.07	-1.02
GLM-Z1-32B	+0.00	-1.28	-1.71	+1.50	+0.28	-1.21
MiMo-7B-RL	-0.01	-1.27	-1.71	+1.70	+0.00	-1.29
DeepSeek-R1-0528	+0.00	-0.55	-1.71	+1.11	+0.28	-0.86
R1D-Llama-70B	+0.00	-0.80	-1.71	+1.40	+0.10	-1.01
R1D-Qwen-1.5B	-0.07	-1.43	-1.71	+1.66	+0.16	-1.38
R1D-Qwen-32B	+0.00	-0.85	-1.71	+1.38	-0.02	-1.19
R1D-Qwen-7B	+0.00	-1.40	-1.71	+1.32	+0.01	-1.78
Gemini-2.5-Flash	-0.48	-0.03	-1.71	+0.54	+0.35	-1.33
Gemini-2.5-Flash-lite	-0.00	-1.45	-1.71	+1.66	+0.75	-0.75
Gemini-2.5-Pro	-0.35	-0.03	-1.71	+0.87	+0.40	-0.81
Phi-4-mini-reasoning	-0.00	-1.47	-1.71	+1.53	+0.30	-1.34
Phi-4-reasoning	-0.35	-0.80	-1.71	+0.81	+0.35	-1.70
Phi-4-reasoning-plus	-0.53	-0.80	-1.71	+0.93	+0.40	-1.71
GPT-5	-0.11	-0.08	-1.71	+0.60	+0.12	-1.18
GPT-5-mini	-0.10	-0.21	-1.71	+0.45	+0.19	-1.39
GPT-5-nano	+0.00	-1.25	-1.71	+0.90	+0.56	-1.49
GPT-OSS-120B-high	-0.08	-0.60	-1.71	+0.58	+0.46	-1.34
GPT-OSS-120B-low	+0.00	-1.05	-1.71	+1.01	+0.16	-1.59
GPT-OSS-120B-medium	+0.00	-0.80	-1.71	+0.58	+0.35	-1.58
GPT-OSS-20B-high	-0.16	-0.69	-1.71	+0.61	+0.46	-1.49
GPT-OSS-20B-low	+0.00	-1.71	-1.71	+2.05	+0.33	-1.04
GPT-OSS-20B-medium	+0.00	-0.96	-1.71	+0.63	+0.38	-1.65
o3	-0.08	-0.10	-1.71	+0.56	+0.10	-1.23
o4-mini	-0.05	-0.27	-1.71	+0.87	+0.17	-0.98
glm-4.5-air	-0.10	-0.54	-1.71	+1.20	+0.16	-0.99

Table 8: Individual standardized CogniLoad contributions: needle fraction from 0.05 to 0.95. Equal nuisance-cell weights are applied within each factor level. Signed log components sum to the net change before rounding.

Setting	Avail.	Cond. correct.	Work	Overhead	Coupling	Net
EXAONE-Deep-32B	+0.00	-0.33	-0.62	+0.65	+0.01	-0.29
QwQ-32B	-0.03	-0.39	-0.62	+0.56	-0.02	-0.50
Qwen3-1.7B	+0.01	-0.27	-0.62	+0.91	-0.32	-0.29
Qwen3-30B-A3B	+0.01	-0.10	-0.62	+0.54	+0.10	-0.07
Qwen3-32B	+0.00	-0.23	-0.62	+0.52	-0.03	-0.36
Qwen3-8B	-0.03	-0.13	-0.62	+0.40	-0.03	-0.41
GLM-Z1-32B	+0.00	-0.25	-0.62	+0.48	+0.05	-0.35
MiMo-7B-RL	+0.00	-0.41	-0.62	+0.63	-0.01	-0.41
DeepSeek-R1-0528	+0.01	-0.01	-0.62	+0.63	-0.02	-0.02
R1D-Llama-70B	+0.00	+0.03	-0.62	+0.64	-0.03	+0.02
R1D-Qwen-1.5B	-0.02	+0.66	-0.62	+0.56	-0.11	+0.47
R1D-Qwen-32B	+0.00	+0.01	-0.62	+0.72	-0.05	+0.06
R1D-Qwen-7B	+0.00	-0.16	-0.62	+0.43	+0.11	-0.24
Gemini-2.5-Flash	+0.01	+0.04	-0.62	+0.70	-0.04	+0.09
Gemini-2.5-Flash-lite	-0.02	+0.36	-0.62	+0.49	-0.22	-0.01
Gemini-2.5-Pro	+0.00	+0.04	-0.62	+0.66	-0.04	+0.05
Phi-4-mini-reasoning	+0.00	+0.07	-0.62	+0.68	+0.05	+0.19
Phi-4-reasoning	+0.03	-0.31	-0.62	+0.72	-0.01	-0.19
Phi-4-reasoning-plus	+0.00	-0.36	-0.62	+0.62	+0.03	-0.33
GPT-5	-0.05	-0.02	-0.62	+0.60	-0.20	-0.29
GPT-5-mini	-0.04	+0.08	-0.62	+0.65	-0.14	-0.06
GPT-5-nano	+0.00	-0.17	-0.62	+0.72	-0.02	-0.09
GPT-OSS-120B-high	-0.00	-0.08	-0.62	+0.67	-0.06	-0.10
GPT-OSS-120B-low	+0.00	-0.22	-0.62	+0.10	-0.01	-0.75
GPT-OSS-120B-medium	+0.00	-0.10	-0.62	+0.62	-0.22	-0.31
GPT-OSS-20B-high	+0.03	-0.17	-0.62	+0.70	-0.06	-0.13
GPT-OSS-20B-low	+0.00	-0.15	-0.62	+0.44	-0.14	-0.47
GPT-OSS-20B-medium	+0.00	-0.05	-0.62	+0.72	-0.10	-0.05
o3	-0.04	-0.01	-0.62	+0.60	-0.34	-0.40
o4-mini	-0.03	+0.07	-0.62	+0.48	-0.21	-0.30
glm-4.5-air	-0.02	-0.02	-0.62	+0.56	-0.01	-0.11

Table 9: Mean signed log contributions across ten tasks within each checkpoint. The three component means sum to the mean log efficiency ratio (before rounding). Ceiling counts identify comparisons for which the observed cost increase forces an efficiency decrease even at perfect high-effort accuracy.

Checkpoint	Avail.	Cond. correct.	Cost	Net	Ceiling
Gemma-4-31B	+0.00	+0.07	-1.53	-1.45	10/10
Qwen3.5-27B	-0.00	+0.05	-1.51	-1.46	10/10
Qwen3.5-9B	-0.01	+0.09	-1.54	-1.46	10/10
GPT-OSS-120B	-0.01	+0.18	-2.36	-2.18	10/10
GPT-OSS-20B	+0.02	+0.24	-3.04	-2.78	10/10
EXAONE-4.0-32B	+0.03	+0.12	-1.90	-1.75	10/10
GLM-4.7-Flash	-0.00	+0.54	-2.12	-1.58	10/10
Nemotron-3-Nano	-0.02	+0.81	-3.58	-2.79	10/10
Granite-4.2-8B	-0.00	+0.07	-0.64	-0.57	8/10

Table 10: All 36 native-task effort cost decompositions. Mean-overhead and coupling contributions sum to the generated-cost contribution. The last column is the high-minus-low-workload change in the log effort-cost ratio.

Task	Checkpoint	Overhead	Coupling	Cost	D_{mn}
CogniLoad	Gemma-4-31B	-1.69	+0.11	-1.58	-0.25
CogniLoad	Qwen3.5-27B	-1.73	+0.38	-1.35	-0.93
CogniLoad	Qwen3.5-9B	-1.11	+0.03	-1.08	-0.18
CogniLoad	GPT-OSS-120B	-2.69	-0.34	-3.04	+1.53
CogniLoad	GPT-OSS-20B	-3.25	-0.54	-3.79	+2.53
CogniLoad	EXAONE-4.0-32B	-1.71	-0.27	-1.97	+5.67
CogniLoad	GLM-4.7-Flash	-1.60	-0.30	-1.90	+0.91
CogniLoad	Nemotron-3-Nano	-7.00	-0.73	-7.73	+1.78
CogniLoad	Granite-4.2-8B	-0.59	-0.73	-1.33	+4.18
GSM8K	Gemma-4-31B	-1.45	-0.02	-1.47	+0.16
GSM8K	Qwen3.5-27B	-1.85	+0.13	-1.72	-0.19
GSM8K	Qwen3.5-9B	-1.97	+0.04	-1.93	-0.15
GSM8K	GPT-OSS-120B	-1.28	+0.01	-1.27	-0.06
GSM8K	GPT-OSS-20B	-2.29	+0.02	-2.27	+0.00
GSM8K	EXAONE-4.0-32B	-1.98	-0.00	-1.99	+0.03
GSM8K	GLM-4.7-Flash	-1.20	-0.03	-1.23	-0.08
GSM8K	Nemotron-3-Nano	-2.54	+0.03	-2.50	-0.07
GSM8K	Granite-4.2-8B	-1.07	+0.06	-1.01	-0.12
ProofWriter	Gemma-4-31B	-1.44	-0.08	-1.52	+0.34
ProofWriter	Qwen3.5-27B	-1.72	+0.04	-1.68	-0.21
ProofWriter	Qwen3.5-9B	-1.79	+0.06	-1.73	-0.26
ProofWriter	GPT-OSS-120B	-2.33	+0.05	-2.29	-0.08
ProofWriter	GPT-OSS-20B	-2.66	+0.02	-2.64	+0.03
ProofWriter	EXAONE-4.0-32B	-1.23	+0.06	-1.18	-0.25
ProofWriter	GLM-4.7-Flash	-2.21	+0.11	-2.11	-0.52
ProofWriter	Nemotron-3-Nano	-0.73	-0.04	-0.77	+0.46
ProofWriter	Granite-4.2-8B	-0.35	+0.00	-0.34	-0.01
ZebraLogic	Gemma-4-31B	-0.74	+0.07	-0.67	-0.56
ZebraLogic	Qwen3.5-27B	-1.34	+0.30	-1.04	-1.94
ZebraLogic	Qwen3.5-9B	-1.12	+0.16	-0.96	-1.53
ZebraLogic	GPT-OSS-120B	-1.73	-0.25	-1.98	+1.09
ZebraLogic	GPT-OSS-20B	-1.98	-0.20	-2.18	+1.09
ZebraLogic	EXAONE-4.0-32B	-1.14	-0.08	-1.22	+0.27
ZebraLogic	GLM-4.7-Flash	-3.18	-0.26	-3.44	+1.89
ZebraLogic	Nemotron-3-Nano	-3.85	-0.48	-4.33	+2.42
ZebraLogic	Granite-4.2-8B	+0.11	+0.02	+0.12	-0.23

The supporting ceiling calculation uses low-effort accuracy q_{low} and observed cost ratio $c = \mathbb{E}[T_{\text{high}}]/\mathbb{E}[T_{\text{low}}]$. Since high-effort accuracy is at most one,

$$\frac{E_{0,\text{high}}}{E_{0,\text{low}}} \leq \frac{1}{q_{\text{low}} c}. \tag{21}$$

Consequently $q_{\text{low}} c > 1$ forces an efficiency decrease at the observed cost. This holds for 88 of the 89 observed decreases and explains why component patterns provide more information than the direction count alone.

G.4 Solver controls and whole-family reference removal

The reasoning effort reference pool contains all 21 settings. Attempt 0 constructs one shared scale per benchmark. Attempts 1 onward evaluate all configurations on covered questions. Reference and evaluation response identities are disjoint. Mean-overhead rank comparisons use all 21 settings, and workload-profile comparisons use the nine models with reasoning on (high effort for GPT-OSS). The single-response model-comparison population supplies a separate, unsplit corroborating analysis.

The constant-workload control sets $W_i = 1$ on exactly the same covered questions. For each of 999 permutation controls, solver workloads are shuffled among those questions using one permutation shared by all evaluated models. This preserves the workload distribution and shared reference while removing its question-specific alignment. Native and permuted-scale calculations retain identical evaluation responses. Constant and shuffled scales often match native mean-overhead ranks closely, demonstrating the limited sensitivity of this ranking statistic.

For the workload-sensitive control, compute $g_m(W) = \text{mean}_{H(W)} \log \overline{T}_{m,i} - \text{mean}_{L(W)} \log \overline{T}_{m,i}$, then center over the fixed model population: $u_m(W) = g_m(W) - M^{-1} \sum_n g_n(W)$. Pairwise contrasts are differences between these model-specific changes. Native and solver scales define their own endpoint groups on the same covered question population. We report the Pearson correlation of their centered vectors, the magnitude ratio $\|u(W_{\text{solv}})\|_2 / \|u(W_{\text{native}})\|_2$, and their root mean squared difference. A high correlation preserves distinctions between models even when their magnitudes differ. Constant workload defines no distinct low/high groups, so it supplies a rank control, not a workload-response estimate.

Table 11: Native versus solver validation on identical covered questions. Workload gives the question-level Spearman correlation between the native and solver-derived workload values. Rank columns correlate generated-length (constant workload) or solver-overhead rankings with native overhead. Profile alignment and magnitude compare centered workload-response vectors for the nine models with reasoning on (high effort for GPT-OSS). Permutation p-values use Holm adjustment across four tasks.

Task	Covered	Workload	Raw rank	Solver rank	Align.	Magnitude	p
CogniLoad	256/256	0.384	0.970	0.977	0.976	0.67	0.004
GSM8K	254/256	0.503	0.997	0.996	0.583	3.33	0.139
ProofWriter	254/256	0.566	0.995	0.992	0.920	1.20	0.012
ZebraLogic	254/256	0.913	0.979	0.979	1.000	1.05	0.004

For permutation inference, the one-sided Monte Carlo probability is $(1 + \#\{\rho_b \geq \rho_{\text{observed}}\})/1000$. Holm adjustment covers four native tasks within each population. Primary profile alignments exceed controls on CogniLoad, ProofWriter, and ZebraLogic, but not GSM8K. Magnitude ratios show that these scales are not interchangeable.

Whole-family removals exclude both Qwen checkpoints together, both GPT-OSS checkpoints together, and each of the five remaining checkpoints as its own family, including all their settings. Each removal supplies one common scale for every target model. We recompute both native and solver profiles on the questions covered after that removal, using the same held-out evaluation attempts. Table 12 summarizes coverage and profile agreement across all seven removals.

Table 12: Ranges over seven whole-family removals. Each removal constructs one shared solver scale for all evaluated models. Native and solver profiles use the same retained questions.

Task	Covered	Profile alignment	Magnitude ratio
CogniLoad	254–256	0.969–0.982	0.53–0.69
GSM8K	253–254	0.560–0.695	2.29–3.47
ProofWriter	243–254	0.780–0.920	0.57–1.26
ZebraLogic	249–254	1.000–1.000	1.02–1.06

Table 13: Solver-normalized comparison with Gemma-4-31B over six additional benchmarks: MATH-500, GPQA, MMLU-Pro, MuSR, ReClor, and LiveCodeBench. Components are mean task-local log contributions. Their sum exponentiates to the geometric-mean efficiency ratio on the covered evaluation questions.

Checkpoint	Avail.	Cond. correct.	Overhead	Coupling	Ratio
EXAONE-4.0-32B	-0.00	-0.06	-0.65	-0.07	0.46
Qwen3.5-27B	-0.01	-0.00	-0.92	-0.04	0.38
Qwen3.5-9B	-0.01	-0.08	-0.98	-0.06	0.32
Gemma-4-31B	+0.00	+0.00	+0.00	+0.00	1.00
Granite-4.2-8B	-0.00	-0.16	-0.74	-0.12	0.36
Nemotron-3-Nano	-0.02	-0.10	-0.40	-0.13	0.52
GPT-OSS-120B	-0.01	-0.03	-0.43	-0.07	0.58
GPT-OSS-20B	-0.05	-0.06	-1.05	-0.06	0.29
GLM-4.7-Flash	-0.03	-0.08	-0.68	-0.11	0.41

Table 14: MATH-500: solver-normalized comparison on 255 covered questions. Reference attempt 0 is excluded from evaluation. Accuracy, tokens, and the efficiency ratio use this support. Overhead and coupling are signed log contributions relative to Gemma-4-31B.

Checkpoint	Accuracy	Tokens	Overhead	Coupling	Ratio
EXAONE-4.0-32B	0.982	6,403	-0.87	-0.17	0.35
Qwen3.5-27B	0.990	7,242	-1.14	-0.02	0.31
Qwen3.5-9B	0.986	9,079	-1.38	-0.01	0.25
Gemma-4-31B	0.988	2,264	-0.00	-0.00	1.00
Granite-4.2-8B	0.973	4,961	-0.68	-0.11	0.45
Nemotron-3-Nano	0.972	3,295	+0.30	-0.68	0.68
GPT-OSS-120B	0.971	2,473	+0.25	-0.34	0.90
GPT-OSS-20B	0.964	5,400	-0.57	-0.30	0.41
GLM-4.7-Flash	0.979	2,994	+0.03	-0.31	0.75

Table 15: GPQA: solver-normalized comparison on 194 covered questions. Reference attempt 0 is excluded from evaluation. Accuracy, tokens, and the efficiency ratio use this support. Overhead and coupling are signed log contributions relative to Gemma-4-31B.

Checkpoint	Accuracy	Tokens	Overhead	Coupling	Ratio
EXAONE-4.0-32B	0.779	13,757	-1.05	+0.11	0.35
Qwen3.5-27B	0.882	18,665	-1.22	-0.03	0.29
Qwen3.5-9B	0.839	20,284	-1.33	-0.00	0.26
Gemma-4-31B	0.868	5,373	-0.00	-0.00	1.00
Granite-4.2-8B	0.635	17,788	-1.19	-0.00	0.22
Nemotron-3-Nano	0.742	19,795	-1.58	+0.28	0.23
GPT-OSS-120B	0.816	11,792	-0.96	+0.17	0.43
GPT-OSS-20B	0.694	27,199	-1.85	+0.23	0.16
GLM-4.7-Flash	0.727	19,890	-1.42	+0.11	0.23

Table 16: MMLU-Pro: solver-normalized comparison on 316 covered questions. Reference attempt 0 is excluded from evaluation. Accuracy, tokens, and the efficiency ratio use this support. Overhead and coupling are signed log contributions relative to Gemma-4-31B.

Checkpoint	Accuracy	Tokens	Overhead	Coupling	Ratio
EXAONE-4.0-32B	0.851	5,630	-0.40	-0.22	0.51
Qwen3.5-27B	0.892	7,119	-0.78	-0.07	0.42
Qwen3.5-9B	0.847	6,961	-0.76	-0.07	0.41
Gemma-4-31B	0.901	3,042	-0.00	-0.00	1.00
Granite-4.2-8B	0.738	6,969	-0.57	-0.25	0.36
Nemotron-3-Nano	0.808	4,979	-0.64	+0.14	0.55
GPT-OSS-120B	0.844	3,674	-0.38	+0.20	0.78
GPT-OSS-20B	0.791	6,862	-0.95	+0.13	0.39
GLM-4.7-Flash	0.831	6,261	-0.80	+0.08	0.45

Table 17: MuSR: solver-normalized comparison on 186 covered questions. Reference attempt 0 is excluded from evaluation. Accuracy, tokens, and the efficiency ratio use this support. Overhead and coupling are signed log contributions relative to Gemma-4-31B.

Checkpoint	Accuracy	Tokens	Overhead	Coupling	Ratio
EXAONE-4.0-32B	0.708	2,828	-0.27	-0.01	0.74
Qwen3.5-27B	0.734	3,779	-0.61	+0.04	0.57
Qwen3.5-9B	0.696	4,306	-0.74	+0.05	0.48
Gemma-4-31B	0.731	2,153	-0.00	-0.00	1.00
Granite-4.2-8B	0.665	2,900	-0.21	-0.08	0.68
Nemotron-3-Nano	0.654	1,412	+0.55	-0.13	1.36
GPT-OSS-120B	0.702	3,545	-0.58	+0.08	0.58
GPT-OSS-20B	0.680	4,872	-0.88	+0.06	0.41
GLM-4.7-Flash	0.659	3,165	-0.39	+0.00	0.61

Table 18: ReClor: solver-normalized comparison on 255 covered questions. Reference attempt 0 is excluded from evaluation. Accuracy, tokens, and the efficiency ratio use this support. Overhead and coupling are signed log contributions relative to Gemma-4-31B.

Checkpoint	Accuracy	Tokens	Overhead	Coupling	Ratio
EXAONE-4.0-32B	0.944	1,982	-0.30	-0.04	0.67
Qwen3.5-27B	0.989	3,038	-0.75	-0.02	0.46
Qwen3.5-9B	0.967	3,405	-0.88	-0.01	0.40
Gemma-4-31B	0.998	1,406	-0.00	-0.00	1.00
Granite-4.2-8B	0.847	3,588	-0.79	-0.14	0.33
Nemotron-3-Nano	0.882	2,310	-0.43	-0.06	0.54
GPT-OSS-120B	0.944	2,282	-0.38	-0.10	0.58
GPT-OSS-20B	0.896	4,365	-1.04	-0.09	0.29
GLM-4.7-Flash	0.897	2,493	-0.50	-0.08	0.51

Table 19: LiveCodeBench: solver-normalized comparison on 237 covered questions. Reference attempt 0 is excluded from evaluation. Accuracy, tokens, and the efficiency ratio use this support. Overhead and coupling are signed log contributions relative to Gemma-4-31B.

Checkpoint	Accuracy	Tokens	Overhead	Coupling	Ratio
EXAONE-4.0-32B	0.747	20,446	-0.99	-0.08	0.29
Qwen3.5-27B	0.817	22,655	-1.02	-0.16	0.29
Qwen3.5-9B	0.589	20,487	-0.77	-0.30	0.23
Gemma-4-31B	0.871	7,011	-0.00	-0.00	1.00
Granite-4.2-8B	0.729	20,921	-0.96	-0.13	0.28
Nemotron-3-Nano	0.729	17,967	-0.61	-0.34	0.33
GPT-OSS-120B	0.897	18,541	-0.54	-0.43	0.39
GPT-OSS-20B	0.796	29,333	-1.02	-0.41	0.22
GLM-4.7-Flash	0.712	29,737	-1.00	-0.45	0.19

G.5 Cross-task transfer with a fixed model population

Table 20 reports accuracy, generated length, and efficiency correlations, with directly bootstrapped differences between their median task-pair correlations. Each replicate resamples questions independently by task, sharing that task's weights across models, metrics, and every task-pair comparison. Differences are recomputed within each replicate, preserving dependence between metrics and overlapping task pairs. The model population remains fixed. Table 21 gives correlations for availability, conditional correctness, overhead, and coupling. The workload components use only the four annotated tasks.

Table 20: Median cross-task Spearman correlations and directly bootstrapped differences of medians. Each task uses shared question/cluster weights across metrics and fixed models. Fourteen models supply six task pairs. Nine models with reasoning on (high effort for GPT-OSS) supply 45 pairs for outcome metrics.

Population	Metric or difference	Estimate	95% interval
Fourteen models	Accuracy	0.418	[0.390, 0.492]
Fourteen models	Length	0.523	[0.422, 0.626]
Fourteen models	Efficiency	0.738	[0.679, 0.769]
Fourteen models	Efficiency minus accuracy	0.321	[0.220, 0.353]
Fourteen models	Efficiency minus length	0.215	[0.095, 0.305]
Nine models	Accuracy	0.567	[0.450, 0.650]
Nine models	Length	0.567	[0.517, 0.617]
Nine models	Efficiency	0.583	[0.533, 0.650]
Nine models	Efficiency minus accuracy	0.017	[-0.083, 0.150]
Nine models	Efficiency minus length	0.017	[-0.067, 0.100]

Table 21: Median component rank correlations across task pairs, using the same fixed-model bootstrap as the outcome metrics. Native workload components use only the four annotated tasks.

Population	Component	Task pairs	Median	95% interval
Fourteen models	Availability	6	0.450	[0.128, 0.475]
Fourteen models	Conditional correctness	6	0.545	[0.479, 0.600]
Fourteen models	Mean overhead	6	0.587	[0.494, 0.670]
Fourteen models	Coupling	6	0.105	[-0.095, 0.297]
Nine models	Availability	45	0.587	[0.356, 0.656]
Nine models	Conditional correctness	45	0.583	[0.400, 0.633]
Nine models	Mean overhead	6	0.642	[0.525, 0.725]
Nine models	Coupling	6	0.550	[0.225, 0.759]

H Estimation checks and additional results

The analyses use fixed response-level records containing question identities, model settings, correctness, generated-token counts, and the available answer-validity and stopping labels. Each estimate uses the question population and generation protocol specified for its experiment.

We check the uniqueness of response identities within each collection, positive recorded costs, the implication $S \leq A$ where acceptance is present, common instance support for paired decompositions, and the accounting identities at numerical precision. Unknown metadata is preserved as unknown. Native workload is joined by question identity. It is not estimated from the target model's own response length.

H.1 Acceptance and stopping in the effort comparisons

Table 22 cross-tabulates answer validity and stopping reasons in the reasoning effort experiment. These independently recorded labels distinguish valid answers that reach the length limit from unavailable answers that stop normally.

Scorer outcome	Length stop	Native normal stop
Rejected ($A = 0$)	3,391	4,319
Accepted ($A = 1$)	277	589,757

Table 22: Independently recorded answer acceptance and native stopping across all 597,744 reasoning effort responses. Of the 277 accepted length-stopped responses, 46 are correct. A further 4,319 responses stopped normally and were rejected by the scorer.

Table 23: Summary of the 90 reasoning effort contrasts visualized in Fig. 8. Counts give the direction of observed changes. Ratios compare higher with lower effort, with the median taken across ten tasks for each model.

Checkpoint	Acc. increases	E_0 increases	Median token ratio	Median E_0 ratio
Gemma-4-31B	10/10	0/10	4.47	0.237
Qwen3.5-27B	10/10	0/10	4.74	0.234
Qwen3.5-9B	10/10	0/10	5.24	0.202
GPT-OSS-120B	10/10	0/10	11.48	0.096
GPT-OSS-20B	10/10	0/10	21.88	0.063
EXAONE-4.0-32B	9/10	0/10	7.23	0.154
GLM-4.7-Flash	10/10	0/10	8.45	0.220
Nemotron-3-Nano	10/10	0/10	31.67	0.110
Granite-4.2-8B	8/10	1/10	1.91	0.576

H.2 Uncertainty and interpretation of comparisons

Reported intervals use 999 paired cluster-bootstrap replicates and the 2.5th and 97.5th percentiles. We first average the recorded attempts within each question and setting, sample question clusters with replacement, and retain all settings and all recorded attempts under the same sampled weights. The generated ratio is recomputed from the weighted correctness and cost moments in every draw. Where the records identify a larger parent cluster, that cluster is sampled as a unit. The reasoning effort LiveCodeBench comparison uses 78 contest clusters. Cluster sizes are retained, so the estimate remains question-weighted rather than giving each differently sized contest equal mass.

Named comparisons use their collection, task, and ordered configuration pair as the key. Cross-task rank comparisons draw shared question weights across the fourteen fixed models within each task and independent weights across tasks. The model population is fixed throughout these comparisons.

The intervals describe variation in the question population conditional on the observed attempts and specified scoring and generation protocols. Selected diagnostic examples use pointwise intervals. The workload comparison families additionally use the simultaneous construction in Section G.2. Cross-task differences use the paired construction for both evaluated model populations in Table 20.

H.3 Complete reasoning effort comparison

The effort summary in Table 23 includes all nine checkpoints on all ten reasoning effort benchmarks. GPT-OSS compares low with high, while the other checkpoints compare off with on. Intermediate settings do not replace these endpoints. We compute every task-local accuracy change, token-cost ratio, and efficiency ratio, and all three outcome contributions. The four native-workload tasks additionally supply 36 four-term contrasts. Each pair uses identical questions and repeat counts. Ratios are computed within tasks before taking the descriptive median across tasks. The reported median efficiency ratio need not equal a ratio of separately reported median accuracy and cost.

The 90 directional counts summarize point estimates. The Granite/ZebraLogic example is selected from this complete comparison to illustrate an exception. Its pointwise interval uses the same unadjusted construction as the other examples. Its paired bootstrap uses the same cluster scheme described above.

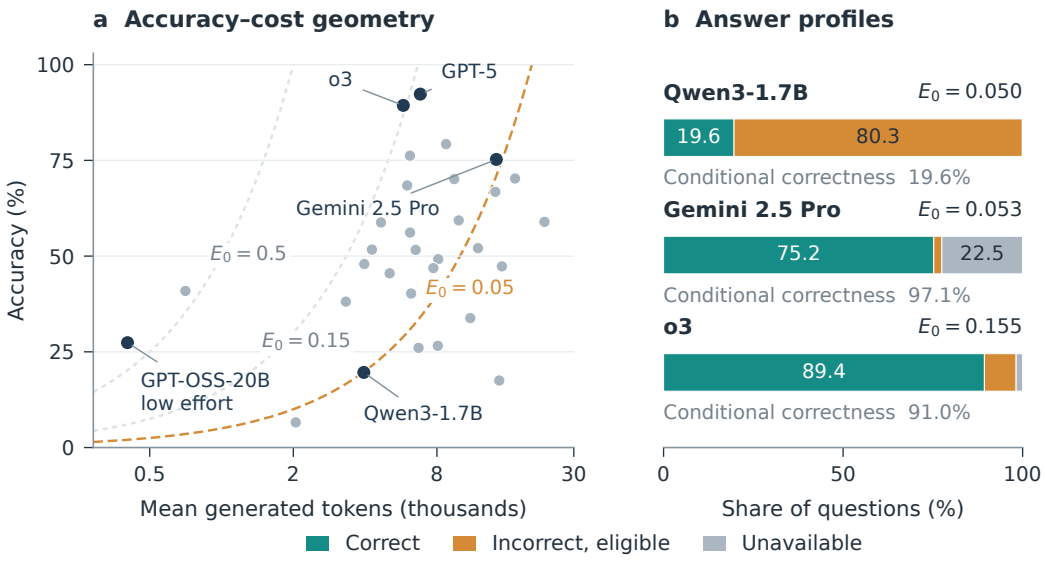

Figure 6: Motivating CogniLoad comparison across 31 settings. Accuracy and mean cost locate models relative to constant-efficiency curves. The outcome profile distinguishes Qwen3-1.7B, Gemini 2.5 Pro, and o3. Their matched-question comparisons are described in the text.

Table 24: Solver-workload calibration. Coverage counts questions with a successful reference. ρ is Spearman correlation between native and solver-derived mean overhead on that covered set. Model comparisons reuse single responses. Reasoning effort comparisons reserve reference attempts.

	Model comparison		Reasoning effort	
Benchmark	Coverage	ρ	Coverage	ρ
CogniLoad	1,312/1,325	0.912	256/256	0.977
GSM8K	1,306/1,319	0.938	254/256	0.996
ProofWriter	2,839/3,000	1.000	254/256	0.992
ZebraLogic	947/1,000	0.996	254/256	0.979

H.4 Aggregate views and motivating examples

Similar token efficiency conceals different answer profiles (Fig. 6). Qwen3-1.7B and Gemini 2.5 Pro achieve $E_0 = 0.0497$ and 0.0530, respectively. Qwen returns an eligible answer on 99.9% of questions but answers only 19.6% of those correctly. Gemini returns eligible answers on 77.5% and answers 97.1% of those correctly, while generating 3.6 times as many tokens per question.

Comparing the models on jointly eligible questions makes the distinction concrete. On 1,300 jointly eligible questions, Qwen solves 258 and o3 solves 1,183. Gemini instead solves 997 of its 1,027 jointly eligible questions, versus 988 for o3. Its nine-answer advantage there becomes an overall deficit of 187 because o3 solves another 196 questions outside that intersection. The accounting distinguishes Qwen's low correctness among eligible answers from Gemini's loss of successful answers through lower availability.

The Phi-4 comparison makes the selection effect visible. The plus variant raises conditional correctness by 0.9 percentage points, but lowers answer availability by 9.1 points and increases mean workload-normalized overhead by 32.3%. Its efficiency is 27.9% lower (ratio 0.72, pointwise 95% interval [0.68, 0.77]). On their 989 jointly eligible questions, however, plus solves 614 versus 647 for the base model. The joint-set comparison reverses the conditional-correctness ordering: the plus variant's higher rate accompanies a smaller, different set of eligible answers. The full 31-setting decomposition is in Section H.

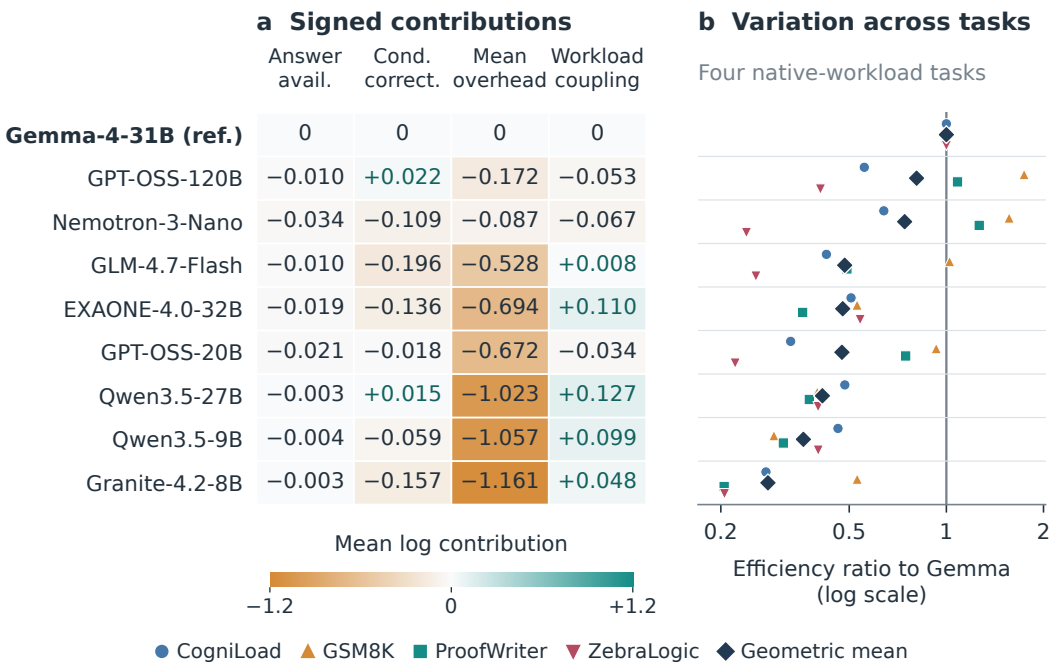

Figure 7: Aggregate efficiency components relative to Gemma-4-31B, averaging task-local log contributions over four native-workload tasks. Individual task ratios show variation around the geometric mean. All models use reasoning on (high effort for GPT-OSS).

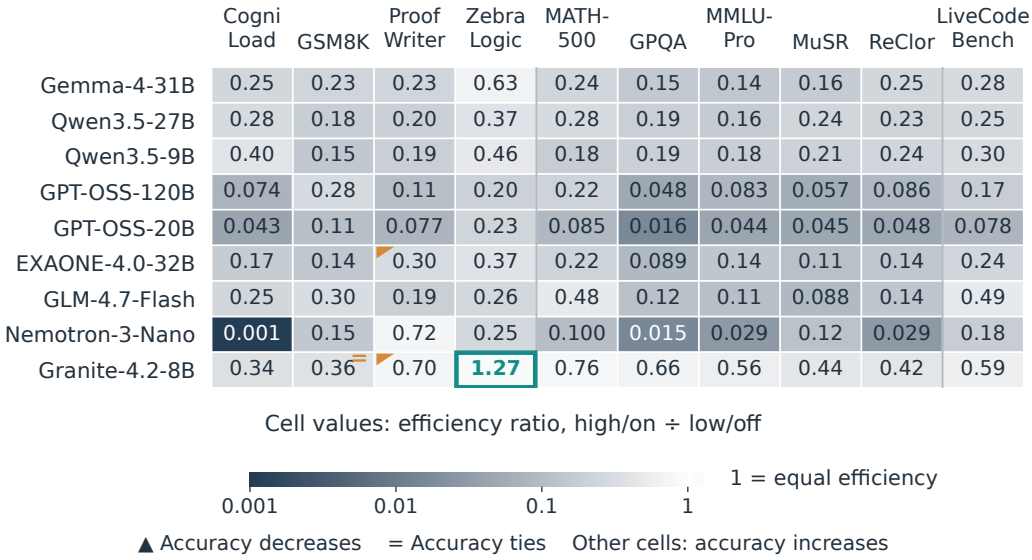

Figure 8: Complete higher/lower-effort efficiency ratios on matched questions with an 81,920-token output cap. Markers identify the two accuracy decreases and one tie. Granite/ZebraLogic improves both accuracy and efficiency. Component patterns are given in Fig. 4 and Table 9.

H.5 Model-comparison profiles and decomposition

The following table reports all 31 CogniLoad settings on the same 1,325 instances. The reference in the corresponding full decomposition is o3. Common question membership gives every configuration the same task and workload distribution.

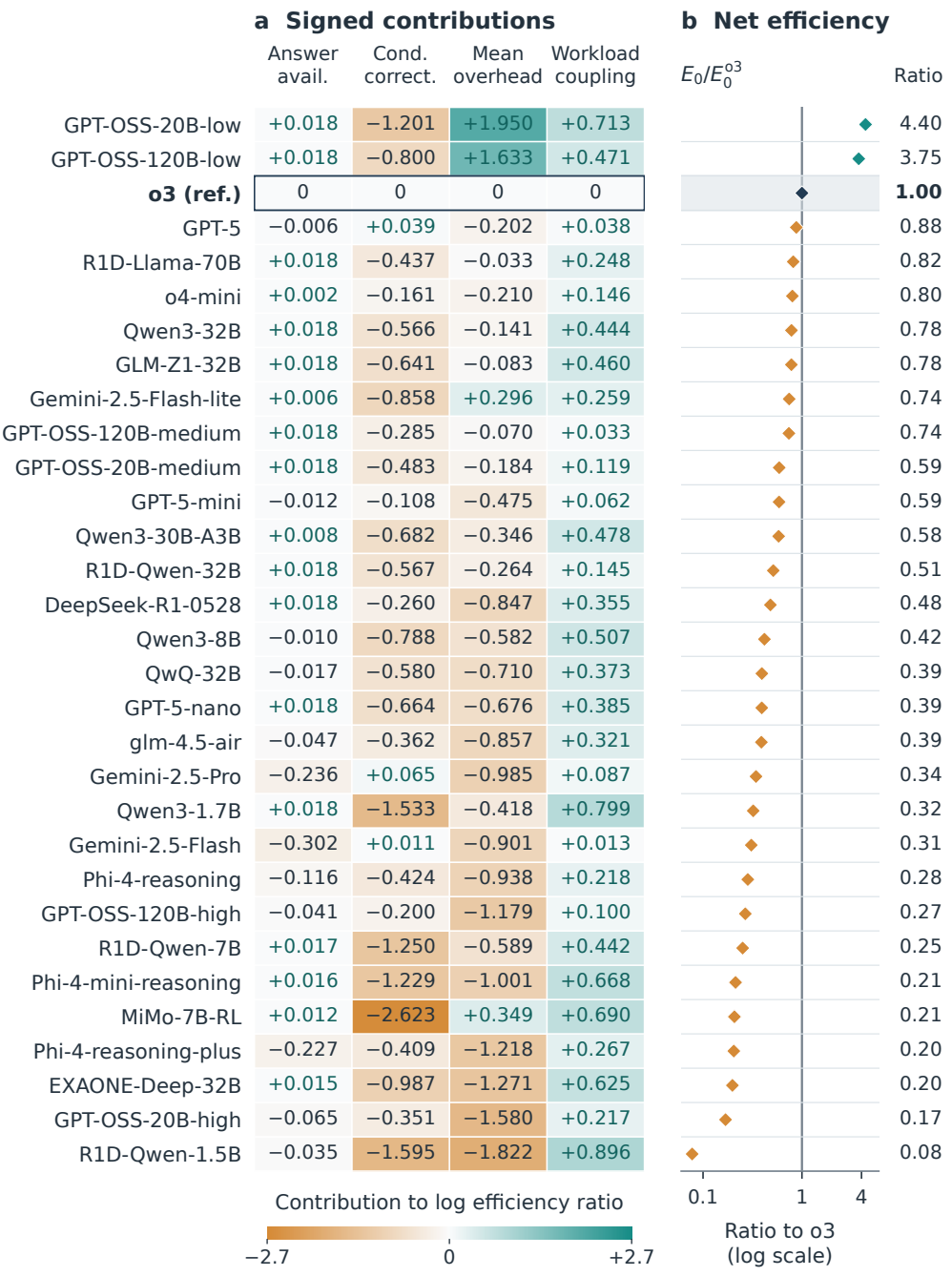

Figure 9: Full 31-configuration CogniLoad workload decomposition on the common 1,325-question model-comparison panel, relative to o3. Cells show signed log contributions from answer availability, conditional correctness, native mean overhead, and workload coupling. Each row sums to the log efficiency ratio. Diamonds and the final column give the corresponding $E_0/E_{0,\mathrm{o3}}$ ratio. Each setting retains its recorded generation protocol.

Table 25: Math/code and competition-mathematics examples. Each coding row has 128 questions $\times$ 64 attempts. Each math row has 30 questions $\times$ four attempts. Rates are fractions. Dashes denote unavailable answer-eligibility metadata.

Task	Setting	Acc.	r_a	r_c	Tokens	E_0
LiveCodeBench	GPT-OSS low	0.725	0.979	0.741	1,154	0.629
LiveCodeBench	GPT-OSS high	0.683	0.728	0.938	8,419	0.081
AIME 2025	o4-mini low	0.617	–	–	2,605	0.237
AIME 2025	o4-mini high	0.917	–	–	13,982	0.066
HMMT Feb. 2025	o4-mini low	0.475	–	–	2,663	0.178
HMMT Feb. 2025	o4-mini high	0.825	–	–	17,637	0.047

Table 26: Different outcome profiles at similar efficiency on the same 1,325 CogniLoad questions. r_a is scorer eligibility. r_c is correctness conditional on eligibility.

Model	Accuracy	r_a	r_c	Mean tokens	E_0
o3	0.894	0.982	0.910	5,783	0.1545
Qwen3-1.7B	0.196	0.999	0.196	3,950	0.0497
Gemini 2.5 Pro	0.752	0.775	0.971	14,196	0.0530

H.6 Cross-task rankings and solver-workload audits

The transfer table compares task-local rankings of the fourteen model-comparison configurations observed on all four native-workload benchmarks. The solver table reports mean-overhead rank agreement and stability under reference removal, using the response splits and common covered sets defined in Section C. Workload-sensitive controls are in Section G.4.

Configuration	Acc.	r_a	r_c	Tokens	E_0	$\overline{\text{VO}}$
GPT-OSS-20B-low	0.274	1.000	0.274	403	0.679	2.9
GPT-OSS-120B-low	0.409	1.000	0.409	706	0.580	3.9
o3	0.894	0.982	0.910	5,783	0.155	20.0
GPT-5	0.923	0.976	0.946	6,813	0.135	24.5
R1D-Llama-70B	0.588	1.000	0.588	4,662	0.126	20.7
o4-mini	0.762	0.984	0.775	6,167	0.124	24.7
Qwen3-32B	0.517	1.000	0.517	4,273	0.121	23.1
GLM-Z1-32B	0.479	0.999	0.480	3,968	0.121	21.8
Gemini-2.5-Flash-lite	0.381	0.988	0.386	3,321	0.115	14.9
GPT-OSS-120B-medium	0.685	1.000	0.685	6,003	0.114	21.5
GPT-OSS-20B-medium	0.562	1.000	0.562	6,172	0.091	24.1
GPT-5-mini	0.792	0.970	0.817	8,740	0.091	32.2
Qwen3-30B-A3B	0.455	0.989	0.460	5,068	0.090	28.3
R1D-Qwen-32B	0.516	1.000	0.516	6,514	0.079	26.1
DeepSeek-R1-0528	0.701	0.999	0.702	9,457	0.074	46.7
Qwen3-8B	0.402	0.972	0.414	6,237	0.064	35.8
QwQ-32B	0.492	0.965	0.510	8,099	0.061	40.7
GPT-5-nano	0.469	1.000	0.469	7,732	0.061	39.4
glm-4.5-air	0.593	0.937	0.633	9,877	0.060	47.2
Gemini-2.5-Pro	0.752	0.775	0.971	14,196	0.053	53.6
Qwen3-1.7B	0.196	0.999	0.196	3,950	0.050	30.4
Gemini-2.5-Flash	0.668	0.726	0.920	14,060	0.048	49.3
Phi-4-reasoning	0.521	0.874	0.596	11,888	0.044	51.2
GPT-OSS-120B-high	0.703	0.943	0.745	17,014	0.041	65.1
R1D-Qwen-7B	0.260	0.998	0.261	6,701	0.039	36.1
Phi-4-mini-reasoning	0.266	0.998	0.266	8,069	0.033	54.5
MiMo-7B-RL	0.066	0.994	0.066	2,047	0.032	14.1
Phi-4-reasoning-plus	0.473	0.783	0.605	14,975	0.032	67.7
EXAONE-Deep-32B	0.338	0.997	0.339	11,041	0.031	71.4
GPT-OSS-20B-high	0.589	0.920	0.641	22,596	0.026	97.2
R1D-Qwen-1.5B	0.175	0.948	0.185	14,600	0.012	123.9

Table 27: Common-question CogniLoad profiles for the model-comparison experiment, with 1,325 responses per configuration. Accuracy, answer availability r_a, and conditional correctness r_c are proportions. E_0 is correct answers per 1,000 generated tokens.

Task pair	Metric	ρ	Lower	Upper
CogniLoad / GSM8K	accuracy	0.587	0.433	0.673
CogniLoad / GSM8K	E0	0.868	0.705	0.903
CogniLoad / GSM8K	VO_native	0.552	0.407	0.754
CogniLoad / ProofWriter	accuracy	0.332	0.268	0.516
CogniLoad / ProofWriter	E0	0.705	0.648	0.736
CogniLoad / ProofWriter	VO_native	0.622	0.525	0.670
CogniLoad / ZebraLogic	accuracy	0.459	0.398	0.495
CogniLoad / ZebraLogic	E0	0.771	0.692	0.829
CogniLoad / ZebraLogic	VO_native	0.864	0.780	0.881
GSM8K / ProofWriter	accuracy	0.304	0.192	0.517
GSM8K / ProofWriter	E0	0.802	0.714	0.881
GSM8K / ProofWriter	VO_native	0.670	0.578	0.793
GSM8K / ZebraLogic	accuracy	0.799	0.636	0.885
GSM8K / ZebraLogic	E0	0.675	0.485	0.745
GSM8K / ZebraLogic	VO_native	0.402	0.191	0.626
ProofWriter / ZebraLogic	accuracy	0.376	0.296	0.464
ProofWriter / ZebraLogic	E0	0.631	0.591	0.675
ProofWriter / ZebraLogic	VO_native	0.389	0.266	0.455

Table 28: Cross-task Spearman correlations across the fourteen shared model-comparison configurations. Entries include pointwise 95% question-bootstrap intervals, keeping the model set fixed. Native overhead uses the benchmark-specific scales in Section B.

Study	Task	Coverage	Native ρ	Removal min.	Top-5 ρ	Retained
Model comparison	CogniLoad	1312/1325	0.912	0.987	0.863	1296
Model comparison	GSM8K	1306/1319	0.938	0.982	0.952	1306
Model comparison	ProofWriter	2839/3000	1.000	0.996	0.991	2801
Model comparison	ZebraLogic	947/1000	0.996	0.996	0.978	888
Reasoning effort	CogniLoad	256/256	0.977	0.978	0.977	256
Reasoning effort	GPQA	194/198	–	0.999	0.984	194
Reasoning effort	GSM8K	254/256	0.996	0.995	0.995	254
Reasoning effort	LiveCodeBench	237/256	–	0.983	0.997	236
Reasoning effort	MATH-500	255/256	–	0.990	0.991	255
Reasoning effort	MMLU-Pro	316/336	–	0.981	0.979	316
Reasoning effort	MuSR	186/192	–	0.996	0.997	181
Reasoning effort	ProofWriter	254/256	0.992	0.994	0.996	251
Reasoning effort	ReClor	255/256	–	0.992	0.996	255
Reasoning effort	ZebraLogic	254/256	0.979	0.996	0.999	254
Competition math	AIME 2025	29/30	–	0.985	0.997	29
Competition math	HMMT Feb. 2025	30/30	–	0.996	0.991	30

Table 29: Solver-workload audits. Coverage is the number of questions with a successful reference before removal. Native ρ compares native and solver-derived mean-overhead ranks. Removal min. is the weakest rank correlation over single-reference deletions. Top-5 removes the five highest-ranked references. Retained is its common covered question count. Every removal correlation holds its retained question set fixed. Model comparison reuses its single recorded responses. Reasoning effort and competition math reserve their first attempt for reference construction.

H.7 REASONING-ENABLED PROFILES

For readability, the following task-level tables show the reasoning-enabled setting of each of the nine reasoning effort checkpoints: high effort for GPT-OSS and thinking on for the other checkpoints. The effort comparisons in Section G.3 also use the lower-effort settings. Each profile reports token costs and accuracy for its specified task and experiment.

Task	Configuration	Acc.	r_a	r_c	Tokens	E_0
CogniLoad	EXAONE-4.0-32B	0.556	0.997	0.558	11,355	0.049
CogniLoad	Qwen3.5-27B	0.928	0.997	0.931	19,816	0.047
CogniLoad	Qwen3.5-9B	0.735	0.995	0.739	16,482	0.045
CogniLoad	Gemma-4-31B	0.904	1.000	0.904	9,346	0.097
CogniLoad	Granite-4.2-8B	0.591	0.995	0.594	22,158	0.027
CogniLoad	Nemotron-3-Nano	0.627	0.975	0.643	10,132	0.062
CogniLoad	GPT-OSS-120B	0.844	0.986	0.856	15,680	0.054
CogniLoad	GPT-OSS-20B	0.768	0.985	0.779	24,128	0.032
CogniLoad	GLM-4.7-Flash	0.581	0.983	0.591	14,129	0.041
GPQA	EXAONE-4.0-32B	0.761	0.987	0.770	13,692	0.056
GPQA	Qwen3.5-27B	0.866	0.996	0.869	18,673	0.046
GPQA	Qwen3.5-9B	0.821	0.992	0.828	20,332	0.040
GPQA	Gemma-4-31B	0.850	1.000	0.850	5,385	0.158
GPQA	Granite-4.2-8B	0.619	0.999	0.620	17,725	0.035
GPQA	Nemotron-3-Nano	0.725	0.951	0.762	19,986	0.036
GPQA	GPT-OSS-120B	0.800	0.994	0.804	11,863	0.067
GPQA	GPT-OSS-20B	0.681	0.908	0.750	27,168	0.025
GPQA	GLM-4.7-Flash	0.713	0.972	0.733	19,660	0.036
GSM8K	EXAONE-4.0-32B	0.961	1.000	0.961	2,351	0.409
GSM8K	Qwen3.5-27B	0.965	1.000	0.965	3,136	0.308
GSM8K	Qwen3.5-9B	0.950	1.000	0.950	4,205	0.226
GSM8K	Gemma-4-31B	0.951	1.000	0.951	1,230	0.773
GSM8K	Granite-4.2-8B	0.931	1.000	0.931	2,274	0.409
GSM8K	Nemotron-3-Nano	0.953	1.000	0.953	787	1.210
GSM8K	GPT-OSS-120B	0.942	0.999	0.943	699	1.347
GSM8K	GPT-OSS-20B	0.944	1.000	0.944	1,313	0.719
GSM8K	GLM-4.7-Flash	0.940	1.000	0.940	1,188	0.791
LiveCodeBench	EXAONE-4.0-32B	0.694	0.987	0.704	21,336	0.033
LiveCodeBench	Qwen3.5-27B	0.759	0.954	0.795	25,031	0.030
LiveCodeBench	Qwen3.5-9B	0.545	0.939	0.580	22,481	0.024
LiveCodeBench	Gemma-4-31B	0.808	1.000	0.808	7,372	0.110
LiveCodeBench	Granite-4.2-8B	0.675	0.999	0.676	22,141	0.030
LiveCodeBench	Nemotron-3-Nano	0.674	0.947	0.712	19,847	0.034
LiveCodeBench	GPT-OSS-120B	0.833	0.953	0.874	21,237	0.039
LiveCodeBench	GPT-OSS-20B	0.737	0.810	0.910	32,185	0.023
LiveCodeBench	GLM-4.7-Flash	0.659	0.842	0.783	32,398	0.020
MATH-500	EXAONE-4.0-32B	0.979	1.000	0.979	6,423	0.152
MATH-500	Qwen3.5-27B	0.987	0.999	0.988	7,303	0.135
MATH-500	Qwen3.5-9B	0.983	0.997	0.986	9,133	0.108
MATH-500	Gemma-4-31B	0.984	1.000	0.984	2,270	0.433
MATH-500	Granite-4.2-8B	0.969	0.998	0.972	5,010	0.193
MATH-500	Nemotron-3-Nano	0.969	0.997	0.972	3,327	0.291
MATH-500	GPT-OSS-120B	0.967	0.994	0.973	2,493	0.388
MATH-500	GPT-OSS-20B	0.960	0.985	0.975	5,411	0.178
MATH-500	GLM-4.7-Flash	0.976	0.997	0.979	3,044	0.321
MMLU-Pro	EXAONE-4.0-32B	0.802	1.000	0.802	5,558	0.144
MMLU-Pro	Qwen3.5-27B	0.839	1.000	0.839	7,246	0.116
MMLU-Pro	Qwen3.5-9B	0.797	1.000	0.797	7,041	0.113
MMLU-Pro	Gemma-4-31B	0.847	1.000	0.847	3,107	0.272
MMLU-Pro	Granite-4.2-8B	0.694	1.000	0.694	6,904	0.101
MMLU-Pro	Nemotron-3-Nano	0.762	0.988	0.771	5,063	0.151
MMLU-Pro	GPT-OSS-120B	0.793	0.993	0.799	3,649	0.217
MMLU-Pro	GPT-OSS-20B	0.745	0.976	0.763	6,779	0.110
MMLU-Pro	GLM-4.7-Flash	0.782	0.996	0.785	6,299	0.124
MuSR	EXAONE-4.0-32B	0.687	1.000	0.687	2,792	0.246
MuSR	Qwen3.5-27B	0.712	1.000	0.712	3,735	0.191

Task	Configuration	Acc.	r_a	r_c	Tokens	E_0
MuSR	Qwen3.5-9B	0.675	1.000	0.675	4,257	0.159
MuSR	Gemma-4-31B	0.707	1.000	0.707	2,123	0.333
MuSR	Granite-4.2-8B	0.645	1.000	0.645	2,850	0.226
MuSR	Nemotron-3-Nano	0.635	0.999	0.636	1,413	0.449
MuSR	GPT-OSS-120B	0.680	1.000	0.680	3,455	0.197
MuSR	GPT-OSS-20B	0.657	0.999	0.658	4,760	0.138
MuSR	GLM-4.7-Flash	0.640	0.999	0.640	3,104	0.206
ProofWriter	EXAONE-4.0-32B	0.713	0.930	0.767	3,703	0.193
ProofWriter	Qwen3.5-27B	0.808	1.000	0.808	4,000	0.202
ProofWriter	Qwen3.5-9B	0.790	1.000	0.790	4,707	0.168
ProofWriter	Gemma-4-31B	0.843	1.000	0.843	1,570	0.537
ProofWriter	Granite-4.2-8B	0.757	0.998	0.758	6,881	0.110
ProofWriter	Nemotron-3-Nano	0.769	1.000	0.769	1,132	0.679
ProofWriter	GPT-OSS-120B	0.947	1.000	0.947	1,628	0.581
ProofWriter	GPT-OSS-20B	0.923	1.000	0.924	2,299	0.402
ProofWriter	GLM-4.7-Flash	0.812	0.985	0.824	3,073	0.264
ReClor	EXAONE-4.0-32B	0.942	1.000	0.942	1,995	0.472
ReClor	Qwen3.5-27B	0.986	1.000	0.986	3,056	0.323
ReClor	Qwen3.5-9B	0.963	1.000	0.963	3,427	0.281
ReClor	Gemma-4-31B	0.996	1.000	0.996	1,421	0.701
ReClor	Granite-4.2-8B	0.844	1.000	0.844	3,594	0.235
ReClor	Nemotron-3-Nano	0.879	0.999	0.880	2,352	0.374
ReClor	GPT-OSS-120B	0.941	1.000	0.941	2,300	0.409
ReClor	GPT-OSS-20B	0.892	1.000	0.892	4,394	0.203
ReClor	GLM-4.7-Flash	0.895	0.999	0.895	2,505	0.357
ZebraLogic	EXAONE-4.0-32B	0.926	1.000	0.926	8,354	0.111
ZebraLogic	Qwen3.5-27B	0.950	0.990	0.959	11,560	0.082
ZebraLogic	Qwen3.5-9B	0.928	0.990	0.937	11,291	0.082
ZebraLogic	Gemma-4-31B	0.907	1.000	0.907	4,420	0.205
ZebraLogic	Granite-4.2-8B	0.833	0.995	0.837	19,772	0.042
ZebraLogic	Nemotron-3-Nano	0.805	0.895	0.900	16,349	0.049
ZebraLogic	GPT-OSS-120B	0.916	0.975	0.939	10,968	0.084
ZebraLogic	GPT-OSS-20B	0.838	0.932	0.899	18,447	0.045
ZebraLogic	GLM-4.7-Flash	0.651	0.994	0.655	12,361	0.053