When LLMs Stop Following Steps: A Diagnostic Study of Arithmetic Procedural Execution in Language Models

Anonymous ACL submission

Abstract

Large language models (LLMs) often achieve strong performance on reasoning benchmarks, but final-answer accuracy alone does not show whether they faithfully execute the procedure specified in a prompt. We introduce a controlled diagnostic benchmark for *procedural execution*, where models are given a step-wise arithmetic procedure and two numeric inputs, and must return the final computed value. Complexity is varied through procedure length and look-back dependencies over intermediate variables. Average first-answer accuracy drops from **63%** on 5-step procedures to **20%** on 95-step procedures. Generation-level analysis shows that failures often involve missing answers, premature answers, self-correction after an initial error and under-executed traces. These findings suggest that apparent reasoning ability can mask substantial weaknesses in faithful long-horizon procedural execution.

1 Introduction

Large language models (LLMs) have shown strong performance on many reasoning tasks, including arithmetic reasoning, symbolic manipulation, multi-step question answering, and code generation (Wei et al., 2022b,a; Weng et al., 2023a; Zhao et al., 2025; Zhou et al., 2022). Recent progress has been driven by scale, instruction tuning, test-time reasoning, and reinforcement learning-based methods that encourage models to generate longer reasoning traces before producing an answer (Guo et al., 2025; Shao et al., 2024).

However, strong final-answer performance does not necessarily reflect faithful execution of the intended procedure. This limitation is especially important for procedure-following tasks, where models must execute explicitly specified multi-step procedures while tracking intermediate states, applying operations in order, and terminating at the intended final output. Prior work has shown that LLMs may rely on shortcut heuristics, shallow patterns, or unstable intermediate computations, and can fail even when the complete procedure is provided in the prompt (Elazar et al., 2022; Shojaee et al., 2025a; Yang et al., 2025b; Xu et al., 2025a). However, existing studies often focus on broad reasoning benchmarks or final-answer outcomes, making it difficult to isolate how procedural execution breaks down when the procedure is fully specified, the elementary operations are simple, and complexity increases primarily through procedure length.

We study *arithmetic procedural execution*: the ability of an LLM to carry out an explicitly specified arithmetic procedure and return the correct final value. We introduce a controlled diagnostic benchmark in which each example provides a step-wise arithmetic procedure and two numeric inputs. The procedure defines a sequence of intermediate variables using basic arithmetic operations, and the model must execute the steps in order to return the final value. Since the reference output is deterministic, this setup isolates procedural failures from ambiguity in task interpretation.

Our benchmark increases complexity along two axes. First, we vary procedure length, testing whether model accuracy is maintained as the number of required steps increases. Second, we introduce look-back dependencies, where a current step may require retrieving and using intermediate variables produced several steps earlier. This stresses procedural state tracking rather than only elementary arithmetic. Across selected combinations of input ranges, data types, operation variants, and look-back settings, the benchmark consists of 55 synthetic datasets, totaling 55,000 examples (see Section 4.1). We evaluate 15 language models across multiple scales and report both answer-level and generation-level metrics, including FAA, CORRECT@ANY, answer position, and step-execution behavior. This paper addresses the following research questions:

- **RQ1:** How does model accuracy change as arithmetic procedures become longer and require deeper look-back dependencies?
- **RQ2:** What generation-level failure modes emerge during arithmetic procedural execution, and how do these failures relate to final-answer accuracy?

Empirically, we find that the average first-answer accuracy (FAA) decreases from **63%** at 5 steps to **20%** at 95 steps. Increasing the dependency depth from using only the immediately preceding intermediate variable to using variables from up to seven previous steps further reduces average accuracy by **23.85** % points, indicating that models struggle not only with longer procedures but also with retrieving and combining non-local intermediate states. Our contributions are: (1) we introduce a controlled synthetic benchmark for evaluating long-horizon arithmetic procedural execution in LLMs released under the CC-BY 4.0 license. (2) we conduct more than 55,000 model evaluations across 15 models to analyze the effects of procedure length and look-back dependency on execution accuracy, and (3) our generation-level analysis reveals that a substantial fraction of failures are not caused by arithmetic inaccuracies alone, but by the model's inability to reliably maintain and update procedural state across multi-step execution. Our benchmark, evaluation code, and generated datasets will be publicly released.[1]

2 Related Work

LLM reasoning has been improved through prompting, scaling, self-verification, and reinforcement learning-based training methods (Wei et al., 2022b,a; Zhou et al., 2023; Weng et al., 2023b; Guo et al., 2025; Shao et al., 2024). We focus on work most related to procedural execution along four directions.

2.1 Algorithmic Execution

Prior work has studied whether neural models can learn and execute algorithmic procedures such as sorting, graph traversal, arithmetic, and program-like computation (Veličković and Blundell, 2021; Xu et al., 2020; Schwarzschild et al., 2021; Weiss et al., 2021; Zhou et al., 2024; Abbe et al., 2023; Kazemnejad et al., 2023). These studies often focus on whether models can learn an algorithm from data and generalize to longer or harder inputs. More recent work evaluates whether reasoning-oriented LLMs can execute explicitly specified procedures at inference time. Shojaee et al. (2025b) show that models can fail on controllable puzzle tasks even when the full solution procedure is provided, while follow-up work finds that performance varies substantially with task design and complexity (Varela et al., 2025). Our work follows this line but studies a simpler and more controlled setting: arithmetic procedures with deterministic ground truth and systematically varied execution length and dependency depth.

2.2 Synthetic Diagnostics

Synthetic diagnostic benchmarks provide controlled settings for testing specific reasoning abilities, such as question answering, compositional instruction following, relational reasoning, and semantic parsing (Weston et al., 2015; Lake and Baroni, 2018; Sinha et al., 2019; Kim and Linzen, 2020). Recent diagnostic studies further show that strong benchmark performance can hide brittle reasoning behavior under small perturbations or increasing complexity (Mirzadeh et al., 2025; Zhou et al., 2025; Kohli et al., 2025). Our benchmark has a similar diagnostic goal, but focuses specifically on procedural execution: the model is given the full procedure in the prompt, and complexity is controlled through procedure length, input type, operation type, and look-back dependency.

2.3 State Tracking

Many reasoning tasks require models to maintain and update intermediate states. Prior work has studied state tracking in procedural text, action-state reasoning, dialogue, and other structured settings (Dalvi et al., 2018; Tandon et al., 2018; Li et al., 2021). Recent work also examines whether LLMs encode intermediate reasoning states or rely on shortcut associations instead (Yang et al., 2025c; Huang et al., 2026; Lu et al., 2025). Our task provides a controlled form of state tracking: procedures define intermediate variables $S_1, S_2, S_3, \ldots$, and later steps may depend on variables produced several steps earlier. By varying look-back depth, we test how well models retrieve and combine non-local intermediate states during execution.

2.4 Compositional Generalization

Compositional generalization studies whether models can recombine familiar parts in new or harder

[1] https://anonymous.4open.science/r/arithmetic-procedural-reasoning-FC16/README.md

configurations (Lake and Baroni, 2018; Kim and Linzen, 2020; Keysers et al., 2020; Hupkes et al., 2020). For Transformers, prior work shows that length generalization and arithmetic generalization depend strongly on task structure and architectural choices such as positional encoding (Zhou et al., 2024; Kazemnejad et al., 2023; Xu et al., 2025b). Our work is related because long procedures require repeated composition of simple operations. However, unlike training-time generalization studies, we test instruction-tuned and reasoning-oriented LLMs at inference time, where the full procedure is explicitly provided. Failures in this setting therefore point to weaknesses in execution, state tracking, or generation behavior rather than lack of access to the rule.

3 Task Formulation

We study whether large language models can faithfully execute an explicitly specified multi-step procedure when the full procedure is provided in the prompt. To isolate procedural execution from broader linguistic ambiguity, we consider a controlled arithmetic setting.

Each example consists of a step-wise natural-language procedure and two numeric inputs, x and y. The procedure initializes $S_1 = x$ and $S_2 = y$, then defines subsequent variables through arithmetic operations over previously computed variables. The model must execute the procedure exactly as written and return only the final numeric result. We vary procedure length to control task difficulty. Formally, for each step $t \geq 3$,

$$S_t = S_i \; o_t \; S_j,$$

where $o_t \in \{+, -, \times, \div\}$ denotes the operation at step t, and S_i, S_j are selected from a bounded history of previously computed variables. The size of this accessible history is controlled by the look-back parameter k, which determines how many earlier intermediate variables remain available for future computation. The final target output is the value computed in the last step of the procedure. Figure 1 shows a representative example of a generated procedure.

This setup gives us full control over procedural complexity while keeping the underlying computation simple and verifiable. Since the correct answer is deterministically computable, the benchmark allows us to distinguish between final-answer success and faithful procedure execution.

```
function(x, y):
    Let S1 = x
    Let S2 = y
    Step 1: S3 = S1 + S2
    Step 2: S4 = S3 - S2
    Step 3: S5 = S4 / S2
    Step 4: S6 = S5 * S3
    Step 5: S7 = S6 - S4
    Final Step: Return S7
```

Figure 1: Representative step-wise arithmetic procedure. Later steps require retrieving and using earlier intermediate variables, making the task a test of procedural state tracking. For example, Step 5 computes S_7 using S_6 and S_4, requiring retrieval of a variable generated two steps earlier (look-back 3).

4 Experimental Setup

We describe the dataset construction, evaluated models, inference procedure, and evaluation metrics used to study procedural execution under increasing complexity.

4.1 Dataset Construction

We construct a controlled benchmark of step-wise arithmetic execution tasks following Section 3. Each instance consists of two numeric inputs, x and y, an explicitly defined arithmetic procedure, and a deterministic reference output. We restrict the benchmark to two input variables to isolate procedural execution and intermediate state tracking from additional complexity due to larger input spaces or variable-management overhead. Models must execute the procedure exactly and return the value from the final step.

The benchmark is designed so that each elementary operation is simple, while the procedural burden increases through procedure length and dependency structure. This allows us to stress-test sustained instruction execution without introducing ambiguity about the correct answer.

Input ranges and data types. We generate inputs from three numeric ranges: [0,1], [1,10] and [10,100]. For the ranges $[1, 10]$ and $[10, 100]$, we include both integer and floating-point inputs. For the range $[0, 1]$, we use floating-point inputs only. This yields five input settings in total: one setting for $[0, 1]$ and two settings each for $[1, 10]$ and $[10, 100]$. Inputs are sampled uniformly at random. Integer values are drawn from the corresponding discrete interval with replacement, while floating-point values are sampled from a continuous interval and rounded to three decimal places.

Operation variants. We construct both single-operation and mixed-operation procedure. In the mixed-operation setting, each procedure may contain all four arithmetic operations. In the single-operation setting, all steps in an procedure use the same operation type. We create four single-operation variants: addition-only, subtraction-only, multiplication-only, and division-only. This design lets us compare homogeneous operation sequences with heterogeneous sequences that require switching between different arithmetic operations. For mixed-operation procedure at each step, arithmetic operators are sampled uniformly from the selected operation set. For each input setting we have five operation variants, four from single and one from mixed.

Procedure Horizons. We vary the number of steps from 5 to 95 in increments of 10 in order to study progressively longer procedural horizons while keeping generation cost and trace analysis computationally manageable. For each execution horizon, we generate 100 examples. Thus, each dataset contains 1,000 examples.

Look-back dependencies. In addition to varying the number of steps, we introduce look-back dependencies to increase the amount of intermediate state the model must track. In the simplest setting, each new variable depends on the immediately preceding intermediate output. In more complex settings, the current step may depend on outputs produced two or three steps earlier. For a look-back setting k, let

$$\mathcal{H}_t^{(k)} = \{S_j \mid \max(1, t-k) \leq j < t\}$$

denote the set of accessible intermediate variables at step t. Two distinct operands,

$$S_i, S_j \sim \mathcal{H}_t^{(k)},$$

are sampled uniformly without replacement, and the next state is computed as

$$S_t = S_i \; o_t \; S_j,$$

where $o_t \in \{+, -, \times, \div\}$. We evaluate look-back settings up to $k = 7$ in order to study progressively longer dependency chains while keeping the benchmark computationally tractable and the generated procedures interpretable.

These settings introduce increasingly non-local dependencies while keeping the ground-truth computation deterministic.

Across the selected combinations of input range, data type, operation variant, and look-back setting, the benchmark contains 55 datasets and 55,000 examples[2].

Reference outputs. For each sample, we compute the reference output using an external deterministic executor. The executor parses the procedure, execute each step in order, and returns a list containing intermediate values of each step and the final output. We intentionally retain edge cases arising during execution, including division-by-zero operations, near-zero numerical instabilities, infinities, and NaN values, in order to evaluate whether models correctly propagate or report such behaviors. The implementation of the executor is provided in Appendix A.

4.2 Model Selection

We evaluate a diverse set of language models spanning different scales, model families, and reasoning-oriented training procedures. The selected models range from small models with approximately 1.5B parameters to large-scale frontier and mixture-of-experts models. For analysis, we group the models into five parameter-scale categories: small (<4B), mid-scale (4B–8B), large (14B–18B), very large (20B–30B), and frontier-scale (>100B). The evaluated models are listed in Table 1.

4.3 Inference Procedure

For each model and benchmark variant, we perform inference on all examples using a fixed prompt format. The prompt contains the system instruction, the full step-wise procedure, and the two input values. The model is instructed to execute the procedure exactly and return the final numeric answer enclosed within `<answer>` and `</answer>` tags. The full prompt template is provided in Appendix B.

After generation, we extract all answer spans matching the required tag format. If the model produces multiple tagged answers, we record both the first extracted answer and the set of all extracted answers. If the output contains no valid tagged

[2]We do not evaluate the full factorial design in order to keep the experimental cost manageable. For **look-back1**, we include all five input settings and all five operation variants, yielding $5 \times 5 \times 1 = 25$ datasets. For the higher look-back settings, we restrict evaluation to the mixed-operation variant and evaluate all five input settings across six additional look-back configurations, yielding $5 \times 1 \times 6 = 30$ datasets. Since each dataset contains 1,000 examples, the full benchmark contains $(25 + 30) \times 1{,}000 = 55{,}000$ examples.

Table 1: Language models evaluated in our experiments.

Abbrev.	Model	Params
Small models (<4B)		
DS-1.5	DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025)	1.5B
QW-4	Qwen3-4B-Thinking-2507 (Yang et al., 2025a)	4B
Mid-scale models (4B–8B)		
OL-7	Olmo-3-7B-Think (Olmo et al., 2025)	7B
MI-7	Mistral-7B-Instruct-v0.3 (Mistral AI, 2024)	7B
DS-7	DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025)	7B
OT-7	OpenThinker2-7B (Guha et al., 2025)	7B
Large models (14B–18B)		
MN-14	Ministral-3-14B-Reasoning-2512 (Liu et al., 2026)	14B
DS-14	DeepSeek-R1-Distill-Qwen-14B (Guo et al., 2025)	14B
QW-14	Qwen3-14B (Yang et al., 2025a)	14B
Very large models (20–30B)		
MG-24	Magistral-Small-2509 (Rastogi et al., 2025)	24B
SV-30	Sarvam-30B (Sarvam Foundation Models Team, 2026)	30B
NV-30	NVIDIA-Nemotron-3-Nano-30B (Blakeman et al., 2025)	30B
Frontier-scale models (>100B)		
OA-120	OpenAI-OSS (Agarwal et al., 2025)	120B
DS-685	DeepSeek-V3.2 (DeepSeek-AI, 2025)	685B
KK-1T	Kimi-K2.5 (Team et al., 2026)	1.1T

answer, the prediction is marked as NULL. All extracted answers are compared against the deterministic reference output after rounding to three decimal places.

All experiments were conducted on NVIDIA H200 140GB GPUs using vLLM (Kwon et al., 2023) for efficient batched inference. All models were evaluated using the same answer extraction and scoring pipeline. Each configuration was evaluated using a single inference run. The complete evaluation required approximately 300 GPU hours.

Decoding configuration. We use *temperature = 0*, *top-p = 1.0*, and a maximum generation length of 32,768 tokens. Generation is terminated upon emitting the model's end-of-sequence (EOS) token or reaching the maximum token limit. For cases where the model reaches the maximum generation length without producing a correct answer, we rerun inference with the model's maximum supported context length.

4.4 Evaluation Metrics

We, evaluate five complementary metrics: First-Answer Accuracy (FAA), Correct@Any (CAA), Parseable Answer Rate (PAR), Answer Position (AP), and Step-Following Behavior (SFB).

Let Y_n denote the reference output for example n, and let $\hat{Y}_n^{(i)}$ denote the i-th answer produced by the model during iterative re-evaluation within a single generation. We define $r(x) = \text{round}(x, 3)$, and let $\mathbf{1}[\cdot]$ denote the indicator function, which evaluates to 1 when the condition is true and 0 otherwise.

First-answer accuracy (FAA). FAA measures whether the first extracted answer is correct:

$$\text{FAA} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{1}\left[r(\hat{Y}_n^{(1)}) = r(Y_n)\right]$$

Correct@Any (CAA). CAA marks an example as correct if any extracted answer matches the reference output:

$$\text{CAA} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{1}\left[\exists i \; r(\hat{Y}_n^{(i)}, 3) = r(Y_n, 3)\right].$$

Parseable Answer Rate (PAR). PAR measures the percentage of examples containing at least one parseable `<answer>` span extracted using regular-expression-based parsing.:

$$\text{PAR} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{1}\left[|\mathcal{A}_n| > 0\right],$$

where $\mathcal{A}_n$ denotes the set of extracted answer spans for example n.

Answer Position (AP). AP measures the normalized position of the first parseable `<answer>` span in the generated output. Let p_n denote the token index of the first parseable answer span and L_n denote the total output length. We compute:

$$\text{AP} = \frac{1}{N}\sum_{n=1}^{N} \frac{p_n}{L_n} \times 100.$$

Step-Following Behavior (SFB). For outputs containing visible reasoning traces, we extract generated step counts by parsing occurrences of patterns of the form `Step k`. Let G_n denote the number of parsed generated steps and T_n denote the expected number of steps. We classify outputs as under-execution when $G_n < T_n$, exact execution when $G_n = T_n$, and over-execution when $G_n > T_n$. We report the percentage of examples in each category.

5 Results

5.1 Performance Degradation with Procedural Complexity

5.1.1 Effect of Execution Horizon

We first analyze how FAA changes as the procedure length increases from 5 to 95. Across models

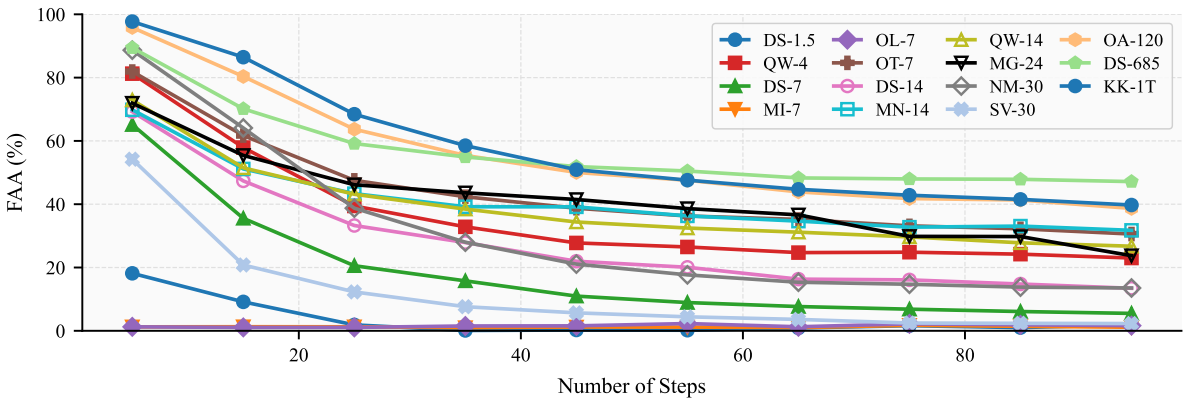


Figure 2: FAA of various language models as a function of Procedure step count (5–95). Performance consistently declines with increasing steps across all models, highlighting growing difficulty in maintaining correct execution over longer procedural sequences despite the simplicity of individual operations.

and experimental settings, we observe a consistent decline in FAA as the number of steps increases (see Figure 2). Averaged across models, the difference between 5-step and 95-step procedures is approximately 43%. NM-30, DS-7, QW-4, KK-1T, OA-120, and DS-14 exhibit the largest degradation, each dropping by more than 55 percentage points. In contrast, MI-7 and OL-7 show minimal decline; however, both models also maintain very low FAA across all procedure lengths, leaving limited room for further degradation. These results indicate that even though each individual operation is elementary, models struggle to sustain correct execution over longer procedures.

Figure 14 in Appendix D further compares FAA and CAA across models. We observe a mean difference of 3.35×10^{-4} between the two metrics across models and procedure lengths, indicating that models rarely self-correct after producing an incorrect first answer. In most cases, incorrect initial predictions remain incorrect throughout the generation.

5.1.2 Effect of Look-back Dependency

Figure 3 shows the relative performance degradation with increasing dependency depth, normalized by each model's average `look_back1` FAA across procedure lengths. Most models show progressively larger degradation as dependency depth increases, suggesting growing difficulty in retrieving and maintaining non-local intermediate states. OA-120, DS-685, QW-14, MN-14, MG-24, and OT-7 achieve high FAA at shorter procedure lengths but degrade substantially with increasing complexity see Figure 16 in Appendix E. In contrast, MI-7 and OL-7 exhibit minimal degradation largely because both begin with very low baseline FAA.

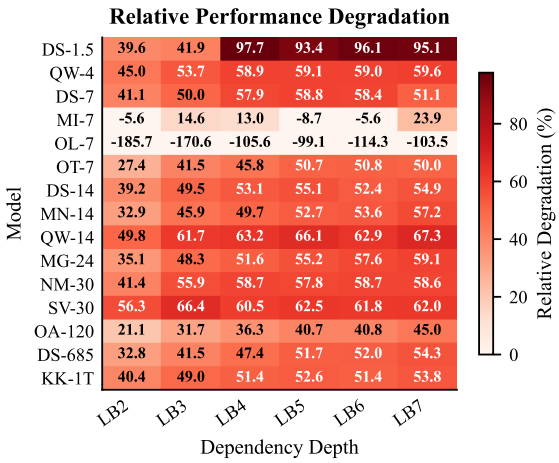


Figure 3: Relative FAA degradation (%) with increasing dependency depth, averaged across procedure lengths and measured with respect to each model's `look_back1` FAA. Each cell shows the percentage drop from the corresponding model's single-step dependency baseline. Larger degradations indicate greater sensitivity to long-range intermediate-state dependencies.

5.2 Input and Task Factors Do Not Fully Explain the Degradation

5.2.1 Effect of Input Range

Figure 4 reports FAA across the three input ranges as a function of procedure length averaged across all models. FAA decreases with increasing number of steps across all ranges. The $[0, 1]$ range generally yields slightly higher FAA than the $[1, 10]$ and $[10, 100]$ ranges, improving by 4.14% on average. Model-wise input-range trends are further shown in Appendix G.2.

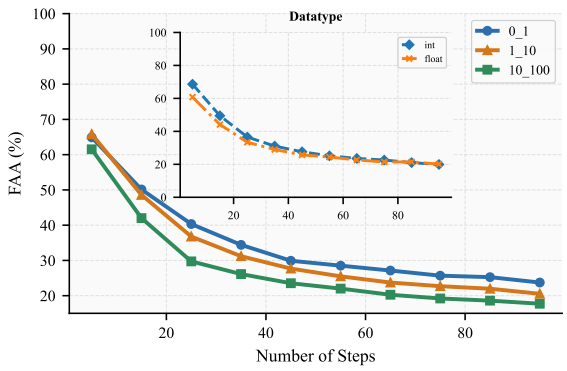


Figure 4: FAA across input ranges as a function of procedure length. The inset compares FAA trends for integer and floating-point input settings.

Figure 8 in Appendix C shows that the median expected value does not consistently increase with procedure length. For the $[0, 1]$ and $[10, 100]$ ranges, correct predictions are generally associated with lower expected-value magnitudes, whereas the $[1, 10]$ range shows a partially reversed trend at higher step counts. These results suggest that differences across input ranges cannot be explained solely by expected-output magnitude.

Figure 15 in Appendix E provides a model-wise heatmap of FAA across input ranges. We observe that some smaller models, such as QW-4 and DS-7, perform relatively better on the $[1, 10]$ range than on the other ranges. This suggests that sensitivity to input range is model-dependent rather than uniform across architectures.

5.2.2 Effect of Data Type

The inset in Figure 4 compares FAA for integer and floating-point inputs. Both data types show a similar downward trend as the number of steps increases, with an average FAA difference of only 2.26% across models and procedure lengths. However, sensitivity varies across models. MN-14, SV-30, DS-14, and QW-14 perform better on integer inputs, with an average FAA gain of 6.53%, while OA-120, KK-1T, and OT-7 achieve higher FAA on floating-point inputs by an average of 6.34%. In contrast, DS-1.5, OL-7, NM-30, and MI-7 show minimal sensitivity to data type. Model-wise datatype trends are further shown in Appendix G.1. Figure 9 in Appendix C further shows that the median expected-output magnitude follows different trends across integer and floating-point settings for correct and incorrect predictions, suggesting that output magnitude alone does not consistently explain model failures.

5.2.3 Effect of Task Type

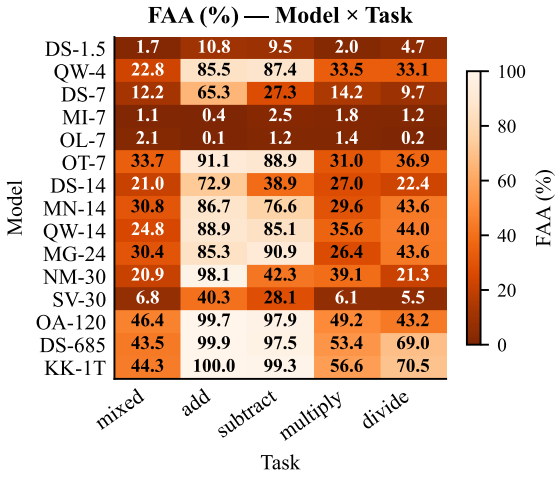


Figure 5: FAA heatmap across models and arithmetic task variants, including addition, subtraction, multiplication, division, and mixed-operation settings.

Figure 5 shows model FAA across task variants. Averaged across models, addition and subtraction outperform multiplication and division by 34.74%, indicating strong operator sensitivity. This gap aligns with the expected-output statistics shown in Figure 10 in Appendix C: multiplication and division produce substantially larger variance and heavier-tailed output distributions than addition and subtraction. For example, multiplication outputs span over 10^{190} in magnitude with unbounded variance, while division exhibits extreme spread due to near-zero denominators, see Table 4 in Appendex H. Such poorly conditioned output distributions can amplify intermediate computation errors and destabilize long-horizon execution. We also observe an average FAA difference of 10.09% between addition and subtraction, suggesting that additive operations may be easier for models to generalize from pretraining data. Mixed-operation tasks further reduce average FAA by 22.06% relative to single-operation settings, suggesting additional difficulty from operator switching across intermediate steps. Model-wise task trends are further shown in Appendix G.3.

5.2.4 Expected-Output Magnitude Analysis

To examine whether numerical scale explains model failures, we compare the median expected output for correct and incorrect predictions across models. Figures 11 to 13 in Appendix C shows that

the relationship between expected-output magnitude and correctness is not uniform across models. NM-30 and DS-7 achieve higher FAA on examples with larger expected values, while DS-1.5, DS-14, and OL-7 show only small differences between correct and incorrect predictions. Overall, expected-output magnitude may contribute to failures in some settings, particularly multiplication and division, but it does not fully explain the broader degradation with increasing procedural complexity. Model-wise median expected-output trends across data types, input ranges, and task variants are further shown in Appendix G.4, Appendix G.5, and Appendix G.6, respectively.

5.3 Generation-Level Failure Modes

5.3.1 Parseable Answer Rate

Table 2 and Figure 18 in Appendix F show PAR as a function of procedure length. Averaged across models, PAR decreases by 24.63% from 5-step to 95-step procedures, indicating increasing difficulty in maintaining structured output generation over longer execution horizons. DS-7, SV-30, MG-24, NM-30, and DS-1.5 exhibit the largest PAR degradation, while OA-120, DS-685, KK-1T, OL-7, and QW-4 remain the most robust across increasing procedure lengths. The consistent decline in PAR with increasing procedure length suggests a strong relationship between long-horizon procedural execution and failures in structured answer generation.

5.3.2 Answer Position

Figure 19 in Appendix F shows the distribution of normalized answer positions across models. OA-120 frequently produces answers very early in the generation, often without explicitly generating intermediate execution steps (see Example I.1 in Appendix I). In contrast, models such as QW-4, NM-30, and SV-30 also exhibit low normalized answer positions, but typically generate all intermediate steps before the first <answer> span. Their low normalized positions arise because the answer often appears within the first 100–200 tokens, followed by very long continuation traces. Thus, low answer position does not necessarily indicate shortcut reasoning or premature execution, but instead reflects differences in generation style and output structure.

5.3.3 Step-wise Execution Breakdown

Table 2 shows the change in exact, under, and over-execution rates between 5-step and 95-step procedures. As procedure length increases, exact execution consistently decreases while under-execution becomes more frequent. DS-7 and QW-4 show the largest shifts, with exact execution dropping by 57.0% and 54.0%, respectively, accompanied by substantial increases in under-execution. In contrast, OA-120 and OL-7 remain comparatively stable across increasing procedure lengths. Although step extraction relies on regex-based parsing of explicit traces, the consistent correlation between increasing procedure length and under-execution suggests that the observed trend cannot be explained solely by parsing artifacts. Instead, these results indicate growing difficulty in maintaining and executing long-horizon procedural state. Model-wise step-following trends are further shown in Appendix G.7

Model	PAR ↓	Exact ↓	Under ↑	Over
OL-7	-4.8	7.0	-7.0	0.0
QW-4	4.4	54.0	-53.8	-0.2
KK-1T	12.1	20.45	-20.45	0
DS-685	12.5	12.9	-12.9	0.0
OA-120	13.3	2.7	-2.7	0.0
MI-7	15.3	34.1	-39.8	+5.7
OT-7	15.5	23.4	-23.4	0.0
QW-14	16.6	17.7	-17.6	-0.1
MN-14	16.7	9.0	-9.0	0.0
DS-14	24.5	16.5	-16.5	0.0
DS-1.5	36.9	32.2	-17.3	-14.9
NM-30	39.9	1.1	-38.9	+37.9
MG-24	42.4	38.1	-38.1	0.0
SV-30	51.0	19.7	-35.9	+16.2
DS-7	60.8	57.0	-45.8	-11.2

Table 2: Difference (%) between 5-step and 95-step procedures. PAR denotes Parseable Answer Rate degradation. Exact, Under, and Over report changes in step-following behavior.

6 Conclusion and Future Work

In this work, we evaluated reasoning models on a controlled benchmark for multi-step procedural execution with varying procedure lengths and look-back dependencies. We find that performance degrades consistently as procedural complexity increases, indicating difficulties in maintaining reliable long-horizon execution and intermediate-state tracking. Future work includes extending the benchmark to branching procedures, symbolic transformations, natural-language workflows, and tool-augmented agents to better understand faithful multi-step instruction execution beyond arithmetic settings.

7 Limitations

Our study is limited to synthetic arithmetic tasks and may not fully generalize to broader reasoning domains. The benchmark does not include branching logic, semantic reasoning, external tool use, or interactive decision-making, and we do not evaluate commercial frontier models. While we observe a strong relationship between procedural complexity and performance degradation, extending the analysis to more diverse structured reasoning tasks remains an important direction for future work.

8 Ethics considerations

Our work evaluates language models on a fully synthetic benchmark for procedural execution using automatically generated arithmetic tasks. Since the dataset is programmatically constructed, it does not contain personally identifiable information (PII), human annotations, or user-generated content. All evaluated models used in this study are publicly available open-source models and were accessed and evaluated in accordance with their respective licenses and intended research use. The benchmark is intended solely for research and evaluation purposes to study procedural reasoning and intermediate state tracking in language models. While the benchmark is designed to isolate procedural reasoning behavior in a controlled setting, it is limited to arithmetic execution and may not fully reflect broader real-world reasoning tasks. We hope this work contributes toward more reliable and transparent evaluation of long-horizon instruction-following behavior in language models.

References

Emmanuel Abbe, Samy Bengio, Aryo Lotfi, and Kevin Rizk. 2023. Generalization on the unseen, logic reasoning and degree curriculum. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31–60. PMLR.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, and 1 others. 2025. Nvidia nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1595–1604, New Orleans, Louisiana. Association for Computational Linguistics.

DeepSeek-AI. 2025. Deepseek-v3.2: Pushing the frontier of open large language models.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Amir Feder, Abhilasha Ravichander, Marius Mosbach, Yonatan Belinkov, Hinrich Schütze, and Yoav Goldberg. 2022. Measuring causal effects of data statistics on language model'sfactual'predictions. *arXiv preprint arXiv:2207.14251*.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, and 1 others. 2025. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Yu Huang, Zixin Wen, Aarti Singh, Yuejie Chi, and Yuxin Chen. 2026. Transformers provably learn chain-of-thought reasoning with length generalization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan, Payel Das, and Siva Reddy. 2023. The impact of positional encoding on length generalization in transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Harsh Kohli, Sachin Kumar, and Huan Sun. 2025. GroundCocoa: A benchmark for evaluating compositional & conditional reasoning in language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8280–8295, Albuquerque, New Mexico. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR.

Belinda Z. Li, Maxwell Nye, and Jacob Andreas. 2021. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827, Online. Association for Computational Linguistics.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, and 1 others. 2026. Ministral 3. *arXiv preprint arXiv:2601.08584*.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. 2025. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183, Albuquerque, New Mexico. Association for Computational Linguistics.

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

Mistral AI. 2024. Mistral-7b-instruct-v0.3. https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3. Accessed: 2026-04-26.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, and 1 others. 2025. Olmo 3. *arXiv preprint arXiv:2512.13961*.

Abhinav Rastogi, Albert Q Jiang, Andy Lo, Gabrielle Berrada, Guillaume Lample, Jason Rute, Joep Barmentlo, Karmesh Yadav, Kartik Khandelwal, Khyathi Raghavi Chandu, and 1 others. 2025. Magistral. *arXiv preprint arXiv:2506.10910*.

Sarvam Foundation Models Team. 2026. Introducing sarvam's sovereign models. https://www.sarvam.ai/blogs/sarvam-30b-105b. Accessed: 2026-03-03.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. 2021. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. In *Advances in Neural Information Processing Systems*.

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

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025a. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *arXiv preprint arXiv:2506.06941*.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025b. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *Preprint*, arXiv:2506.06941.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. 2019. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515, Hong Kong, China. Association for Computational Linguistics.

Niket Tandon, Bhavana Dalvi, Joel Grus, Wen-tau Yih, Antoine Bosselut, and Peter Clark. 2018. Reasoning about actions and state changes by injecting commonsense knowledge. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 57–66, Brussels, Belgium. Association for Computational Linguistics.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, and 1 others. 2026. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*.

Iñaki Dellibarda Varela, Pablo Romero-Sorozabal, Eduardo Rocon, and Manuel Cebrian. 2025. Rethinking the illusion of thinking. In *Artificial Intelligence XLII: 45th SGAI International Conference on Artificial Intelligence, AI 2025, Cambridge, UK, December 16–18, 2025, Proceedings, Part I*, pages 116–129, Berlin, Heidelberg. Springer-Verlag.

Petar Veličković and Charles Blundell. 2021. Neural algorithmic reasoning. *Patterns*, 2(7):100273.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2021. Thinking like transformers.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023a. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023b. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575, Singapore. Association for Computational Linguistics.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *Preprint*, arXiv:1502.05698.

Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken ichi Kawarabayashi, and Stefanie Jegelka. 2020. What can neural networks reason about? In *International Conference on Learning Representations*.

Xingcheng Xu, Zibo Zhao, Haipeng Zhang, and Yanqing Yang. 2025a. Principled understanding of generalization for generative transformer models in arithmetic reasoning tasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4721–4747.

Xingcheng Xu, Zibo Zhao, Haipeng Zhang, and Yanqing Yang. 2025b. Principled understanding of generalization for generative transformer models in arithmetic reasoning tasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4721–4747, Vienna, Austria. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yiran Yang, Haifeng Sun, Jingyu Wang, Qi Qi, Zirui Zhuang, Huazheng Wang, Pengfei Ren, Jing Wang, and Jianxin Liao. 2025b. Unveiling internal reasoning modes in llms: A deep dive into latent reasoning vs. factual shortcuts with attribute rate ratio. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2186–2206.

Yiran Yang, Haifeng Sun, Jingyu Wang, Qi Qi, Zirui Zhuang, Huazheng Wang, Pengfei Ren, Jing Wang, and Jianxin Liao. 2025c. Unveiling internal reasoning modes in LLMs: A deep dive into latent reasoning vs. factual shortcuts with attribute rate ratio. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2186–2206, Suzhou, China. Association for Computational Linguistics.

Eric Zhao, Pranjal Awasthi, and Sreenivas Gollapudi. 2025. Sample, scrutinize and scale: Effective inference-time search by scaling verification. *arXiv preprint arXiv:2502.01839*.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Joshua M. Susskind, Samy Bengio, and Preetum Nakkiran. 2024. What algorithms can transformers learn? a study in length generalization. In *The Twelfth International Conference on Learning Representations*.

Hattie Zhou, Azade Nova, Aaron Courville, Hugo Larochelle, Behnam Neyshabur, and Hanie Sedghi. 2023. Teaching algorithmic reasoning via in-context learning.

Hattie Zhou, Azade Nova, Hugo Larochelle, Aaron Courville, Behnam Neyshabur, and Hanie Sedghi. 2022. Teaching algorithmic reasoning via in-context learning. *arXiv preprint arXiv:2211.09066*.

Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, and Beidi Chen. 2025. GSM-∞: How do your LLMs behave over infinitely increasing reasoning complexity and context length? In *Forty-second International Conference on Machine Learning*.

Appendix Contents

A Expected Output Computation **13**

B Prompt used during Inference **14**

C Results: Expected Output Analysis **14**

D Results: FAA vs CAA **19**

E Results: HeatMap **20**

F Results: Generation-Level Failure Modes **21**

G Results: Model wise Analysis **23**

G.1 FAA Across Data Types 23

G.2 FAA Across Range of Input 28

G.3 FAA across task variants 33

G.4 Median expected output across data types. 38

G.5 Median expected output across input ranges 46

G.6 Median expected output across Task type 61

G.7 Step Analysis across procedure lengths. 76

H Results: Statistical Reliability of Accuracy Estimates **84**

I Model Generations **85**

I.1 Example 1: 85

A Expected Output Computation

```
def exec_algo(prompt, inputs):
    lines = prompt.strip().split("\n")

    vars_dict = {}

    vars_dict["S1"] = inputs[0]
    vars_dict["S2"] = inputs[1]

    for line in lines:
        line = line.strip()

        if "=" not in line:
            continue

        try:
            expr_part = line.split(":")[-1].strip()
            left, right = expr_part.split("=")

            left = left.strip()
            right = right.strip()

            first_var = right.split()[0].strip()
            task = right.split()[1].strip()
            second_var = right.split()[2].strip()
            value = eval(f"vars_dict['{first_var}'] {task} vars_dict['{second_var}']")

            vars_dict[left] = value

        except Exception as e:
            pass

    return [str(v) for v in vars_dict.values()]
```

Figure 6: Algorithm used to evaluate step-wise arithmetic procedures and compute deterministic reference outputs.

B Prompt used during Inference

```
You are a deterministic procedure executor.

You will be given:
1. A procedure (step-by-step instructions)
2. Input values

Your task is to simulate the procedure exactly as written.

Execution rules:
- Execute each step sequentially, without skipping or reordering.
- Maintain exact intermediate state.
- Do not reinterpret or optimize steps.

Behavior:
Show step-by-step execution.
Include intermediate variable values after each step.

Output rules:
- Show all intermediate steps clearly.
- Use the same variable names as in the procedure (S1, S2, etc.).
- That line must be in the format: <answer>FINAL_RESULT</answer>.
- Do not include any additional text before or after these tags.
- Do not use phrases like 'The final result is'.
- The output must contain only the tagged answer.
- Example (correct):
    <answer>42</answer>

Example (incorrect):
The final result is <answer>42</answer>
Answer: 42

Procedure:
function(x, y):
Let S1 = x
Let S2 = y
Step 1: S3 = S1 + S2
Step 2: S4 = S2 * S3
Step 3: S5 = S4 \/ S2
Step 4: S6 = S5 - S2
Step 5: S7 = S2 * S6
Final Step: Return S7

Inputs:
x = 5, y = 8
```

Figure 7: Inference prompt used for procedural execution experiments.

C Results: Expected Output Analysis

This appendix provides a detailed analysis of how the magnitude of expected outputs relates to model performance across different settings. We report the median expected output as a function of the number of steps, separating correct and incorrect predictions to better understand whether numerical scale contributes to errors. These plots complement the main results by showing that while numerical magnitude can influence performance in certain cases (notably multiplication and division), it does not fully account for the observed degradation in accuracy with increasing procedural complexity.

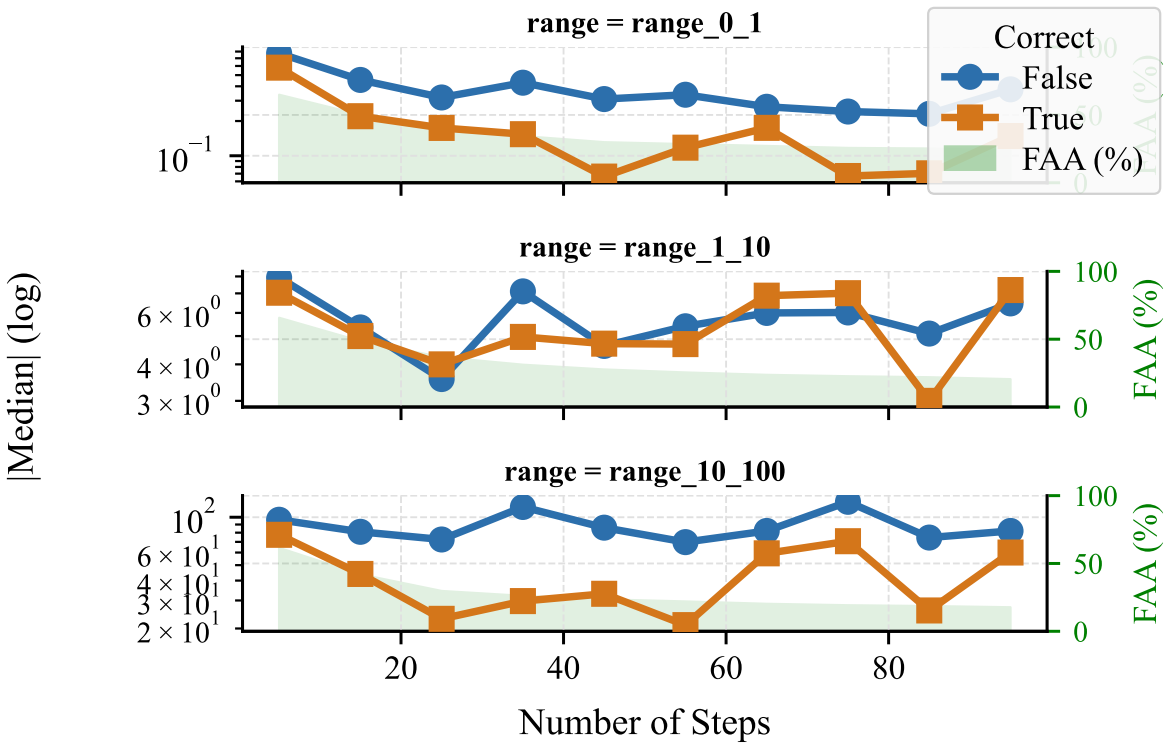


Figure 8: Median expected output across steps for different input ranges ([0,1], [1,10], [10,100]) separated by correct and incorrect model prediction.

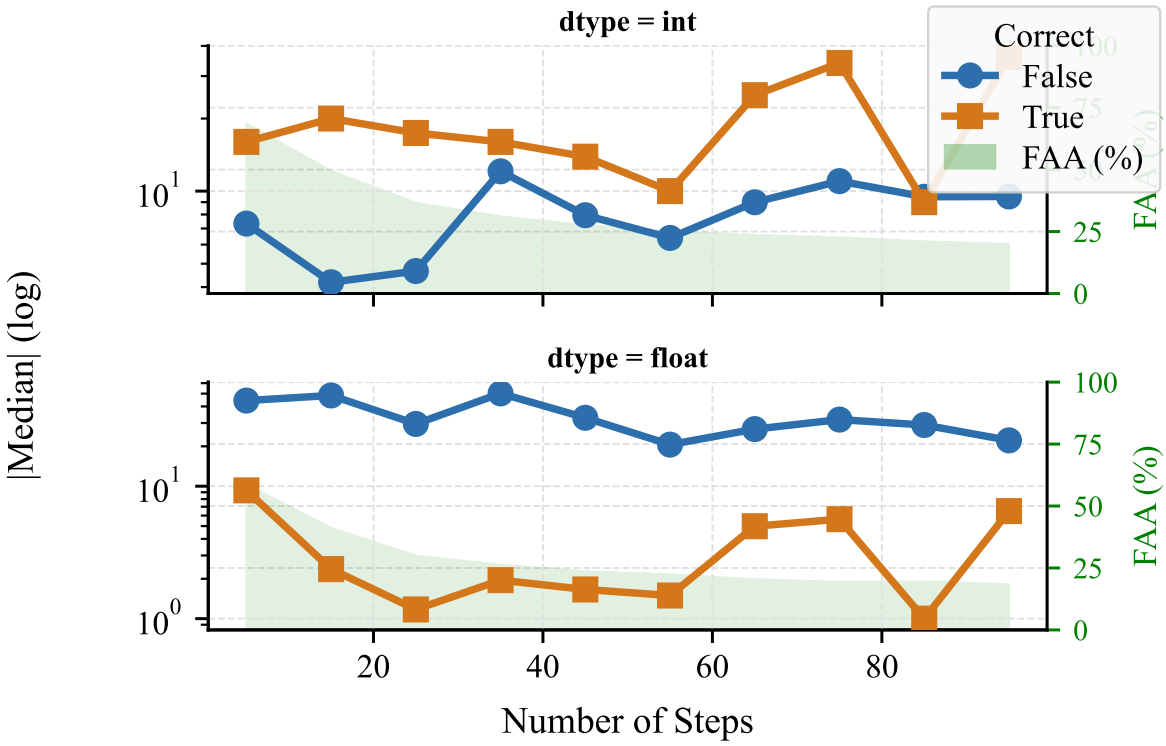


Figure 9: Median expected output across steps for integer and floating-point inputs, separated by correct and incorrect predictions.

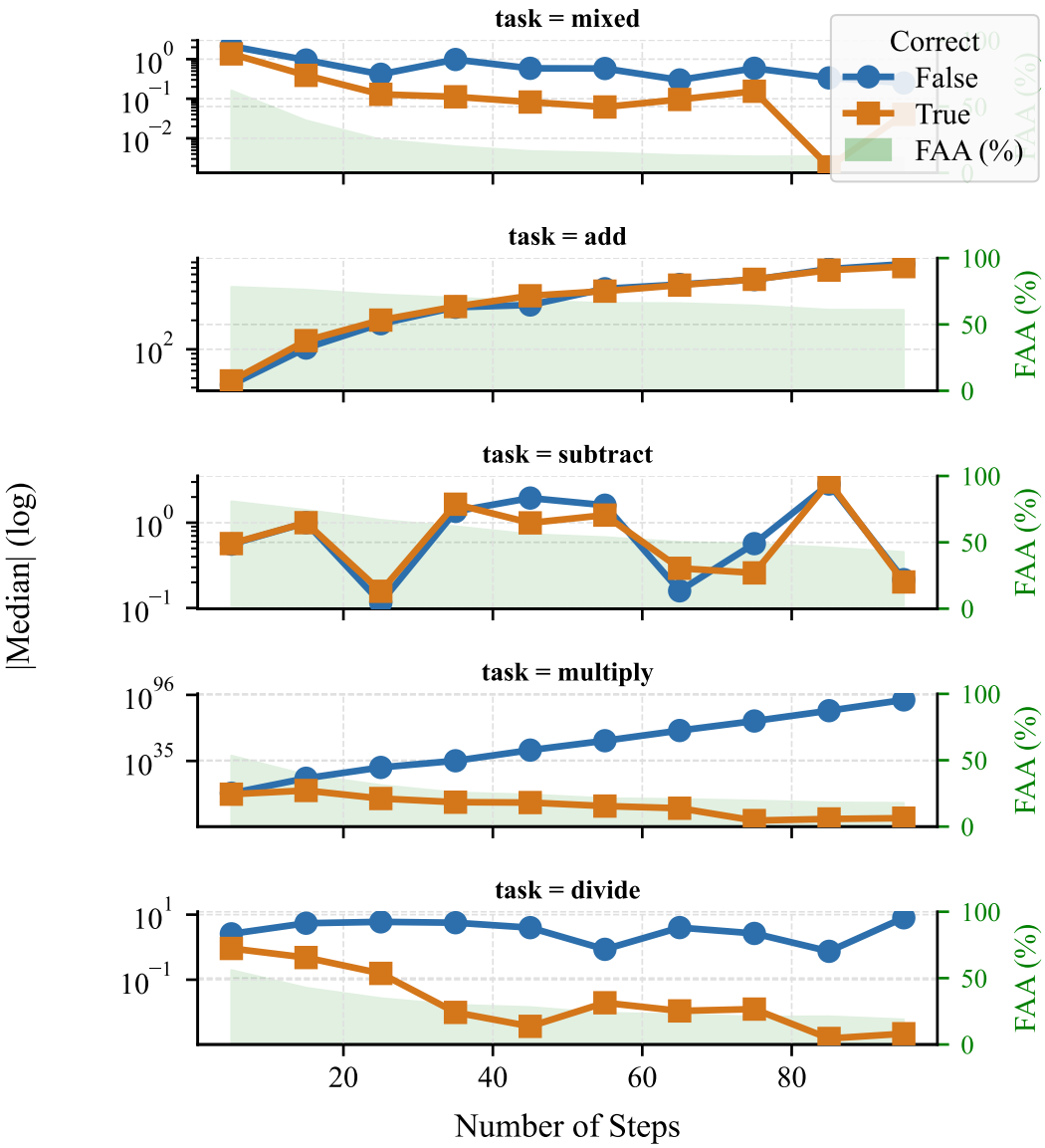
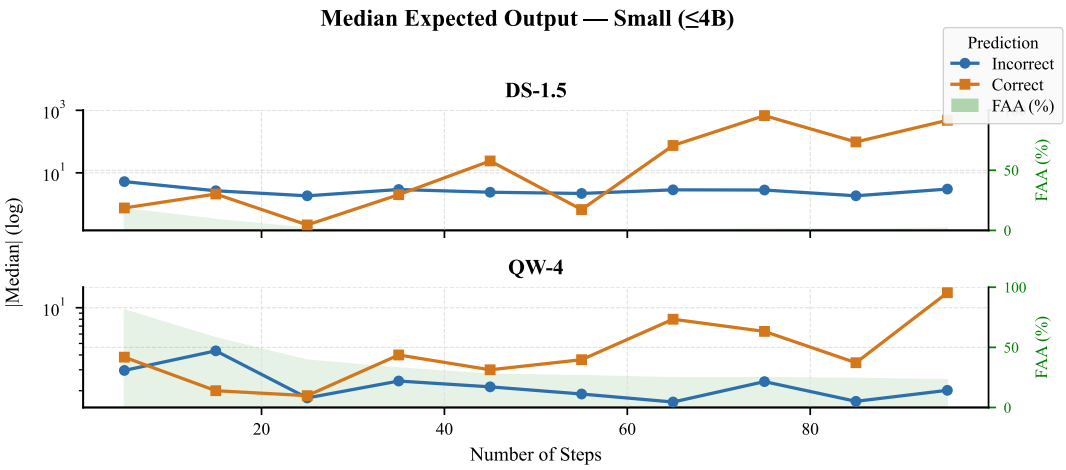
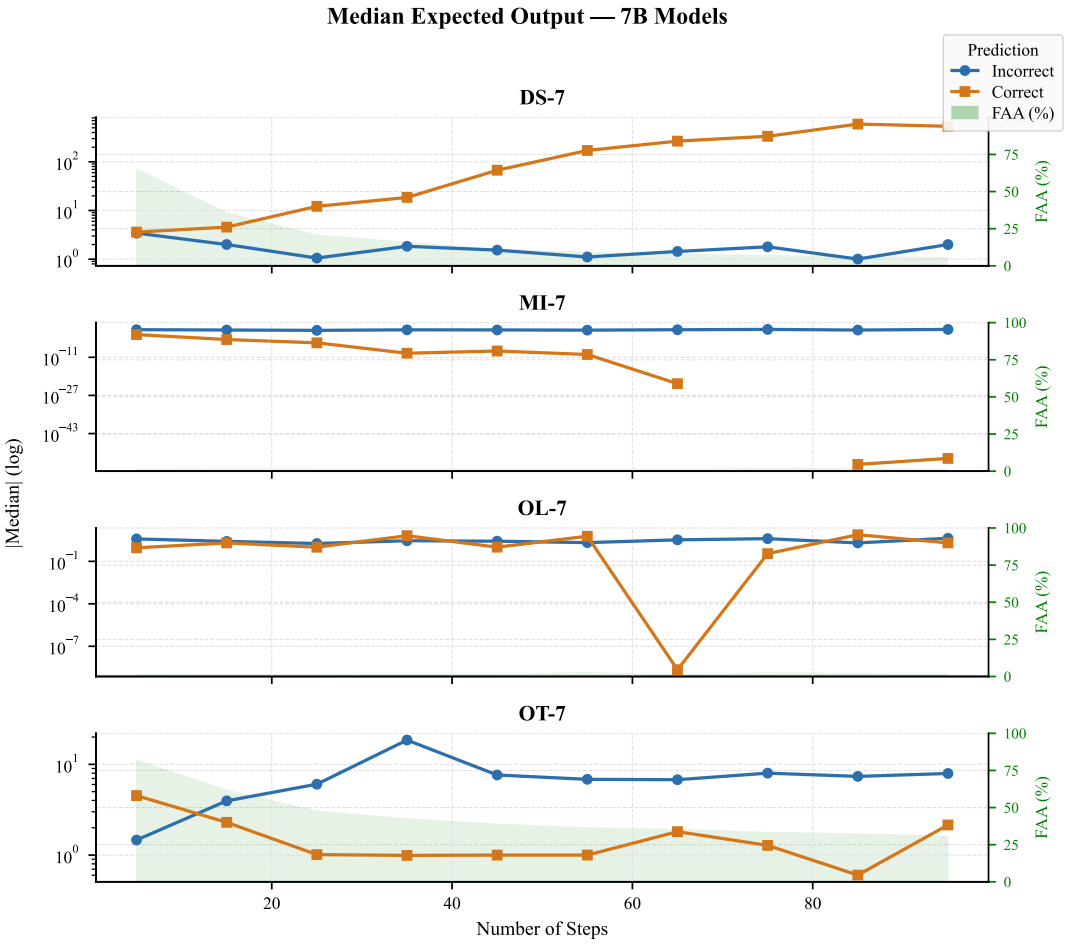


Figure 10: Median expected output across steps for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness. Multiplication and division exhibit a larger separation between correct and incorrect predictions compared to other tasks.

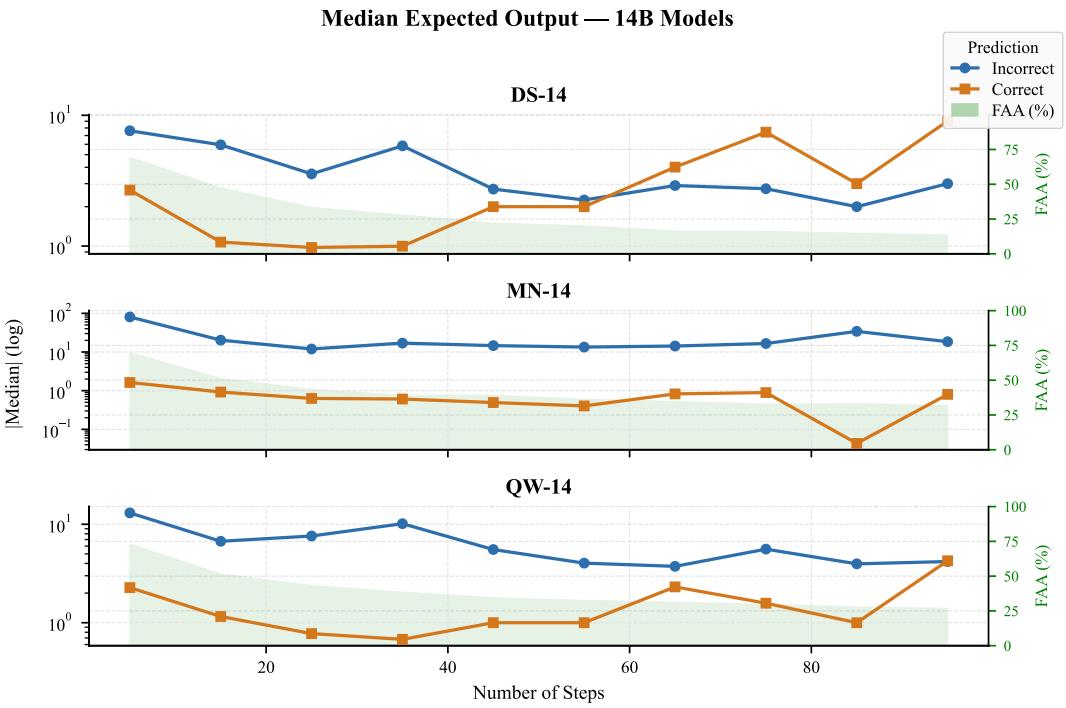


(a) Small models (≤4B)

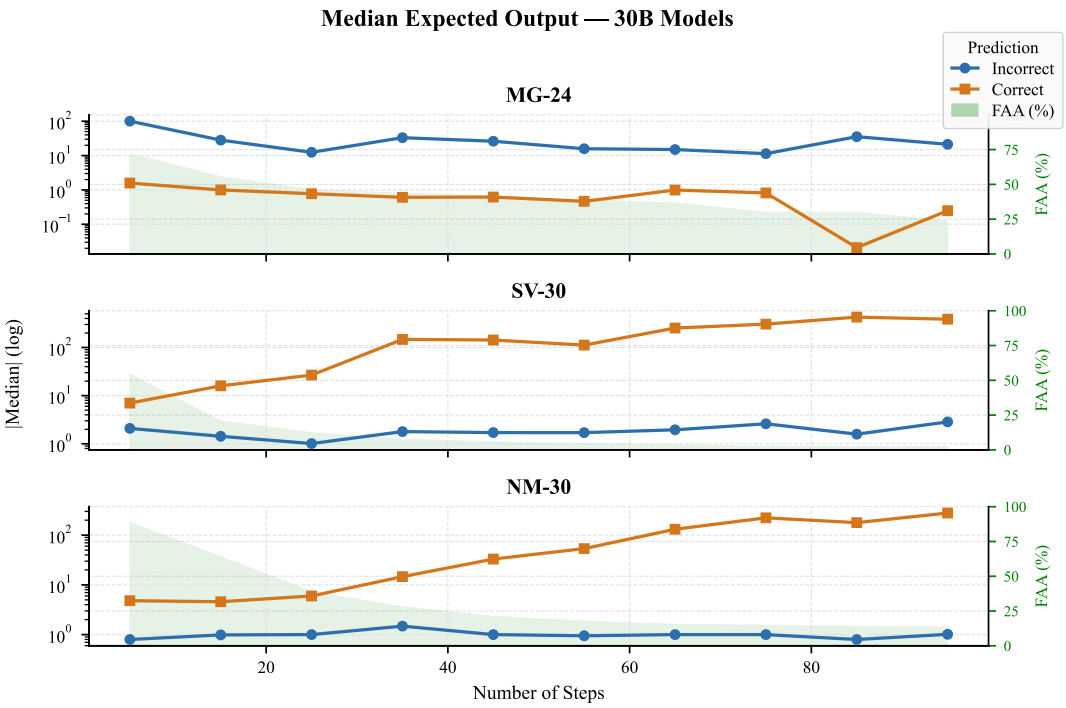


(b) 7B models

Figure 11: Median expected output across steps for small (≤4B) and medium (7B) models. Trends vary across models, with no consistent relationship between output magnitude and correctness.

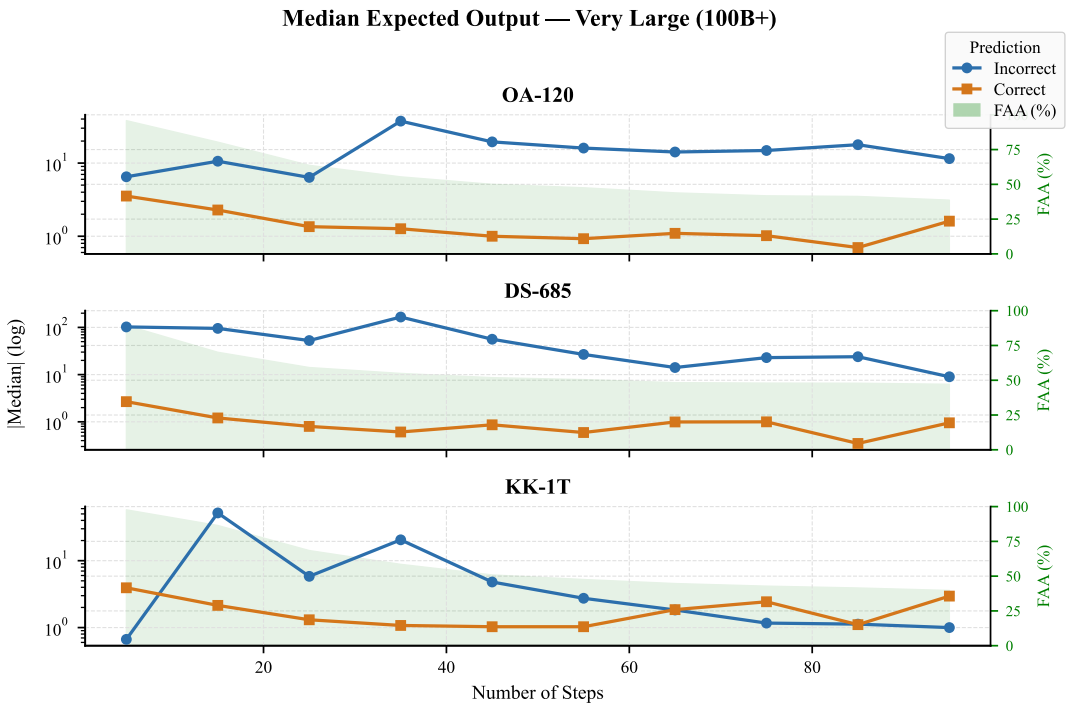


(a) 14B models



(b) 30B models

Figure 12: Median expected output across steps for Mid range models (14B, 30B). While some models show sensitivity to output magnitude, the overall trend indicates that numerical scale alone does not explain performance degradation.



(a) Very large models (100B+)

Figure 13: Median expected output across steps for larger models (≥100B). While some models show sensitivity to output magnitude, the overall trend indicates that numerical scale alone does not explain performance degradation.

D Results: FAA vs CAA

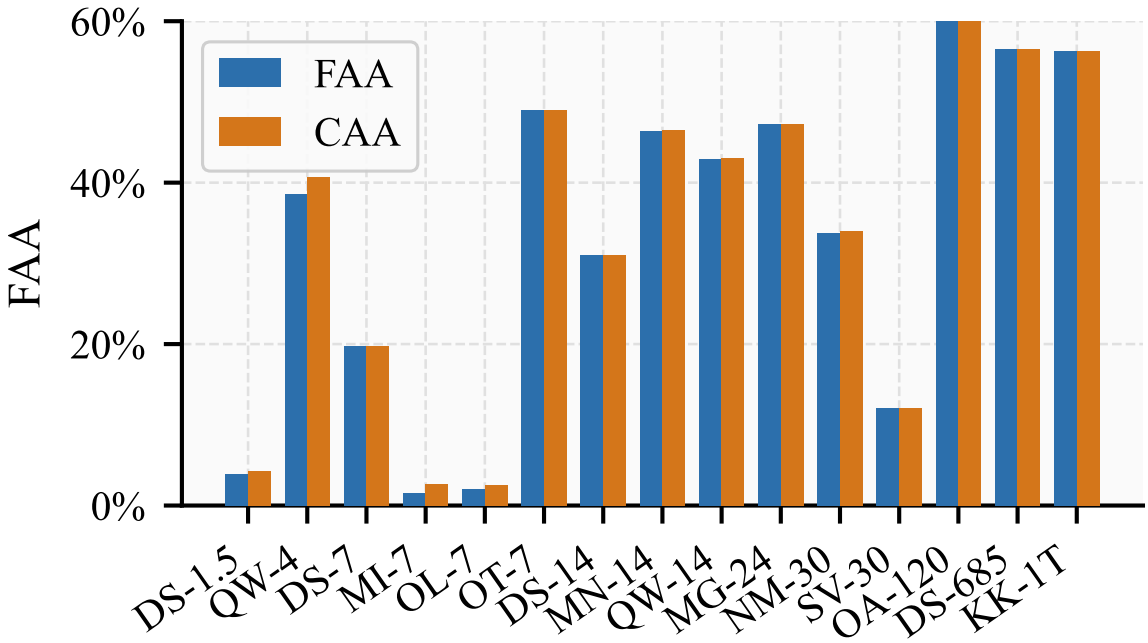


Figure 14: Comparison of FAA and CAA across models.

E Results: HeatMap

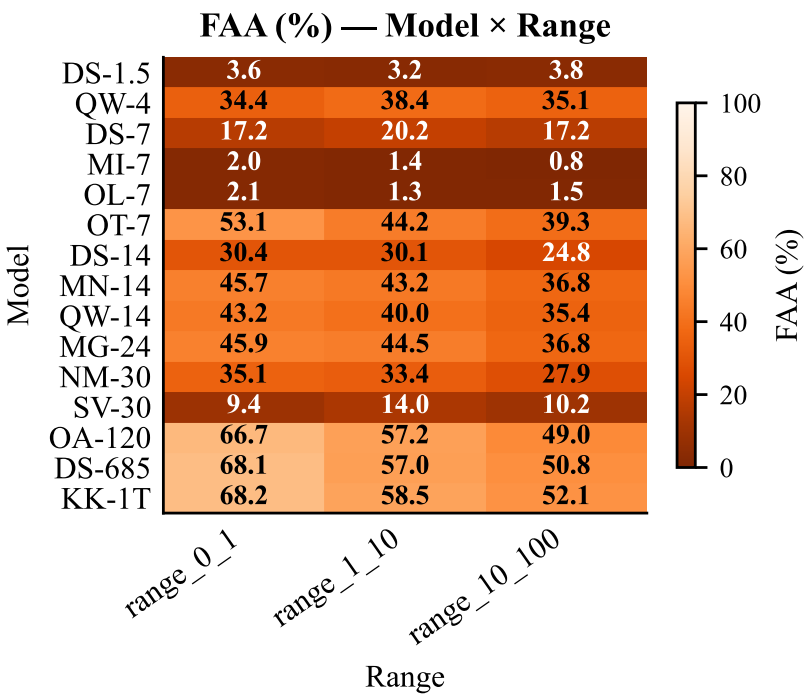


Figure 15: FAA heatmap across models and input ranges. Performance varies across models, with no uniform sensitivity to range; some models achieve higher FAA in intermediate ranges ([1,10]) than in smaller or larger ranges.

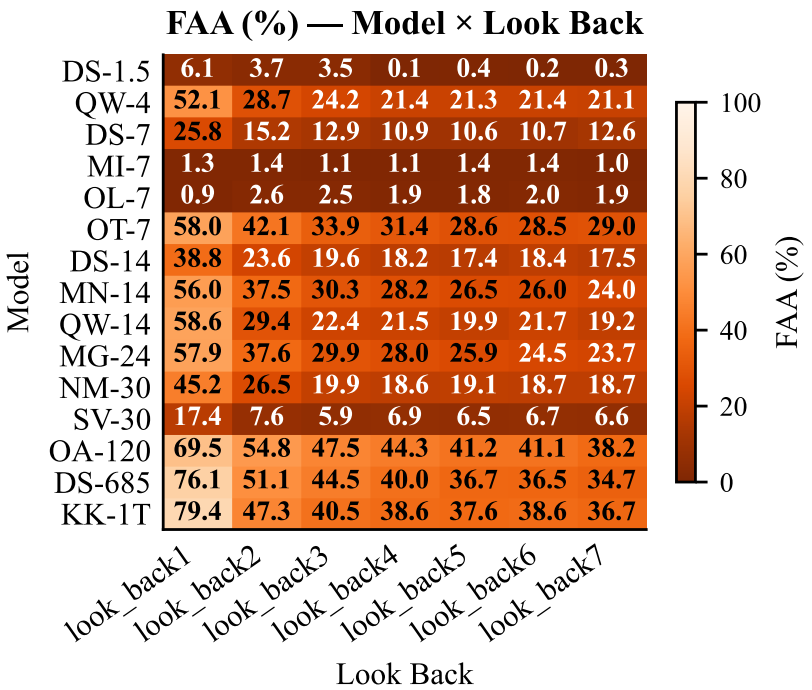


Figure 16: FAA heatmap across models and look-back dependency.

F Results: Generation-Level Failure Modes

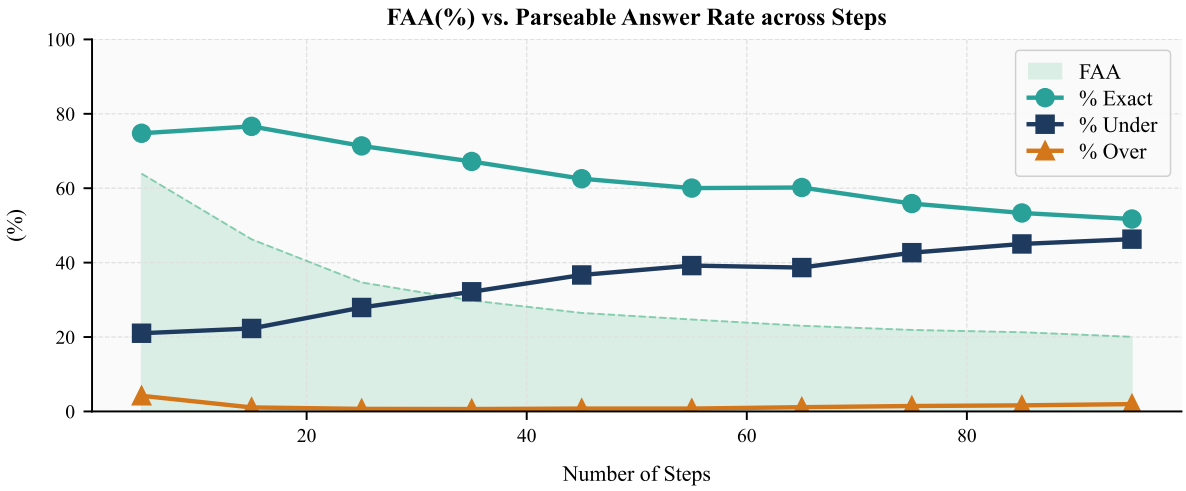


Figure 17: FAA and execution behavior across increasing algorithm lengths. While exact-match FAA (dashed) steadily declines, the proportion of exact executions decreases and under-executed generations rise, with over-execution remaining minimal. This shift indicates that errors at higher step counts are primarily driven by incomplete procedural execution rather than incorrect arithmetic.

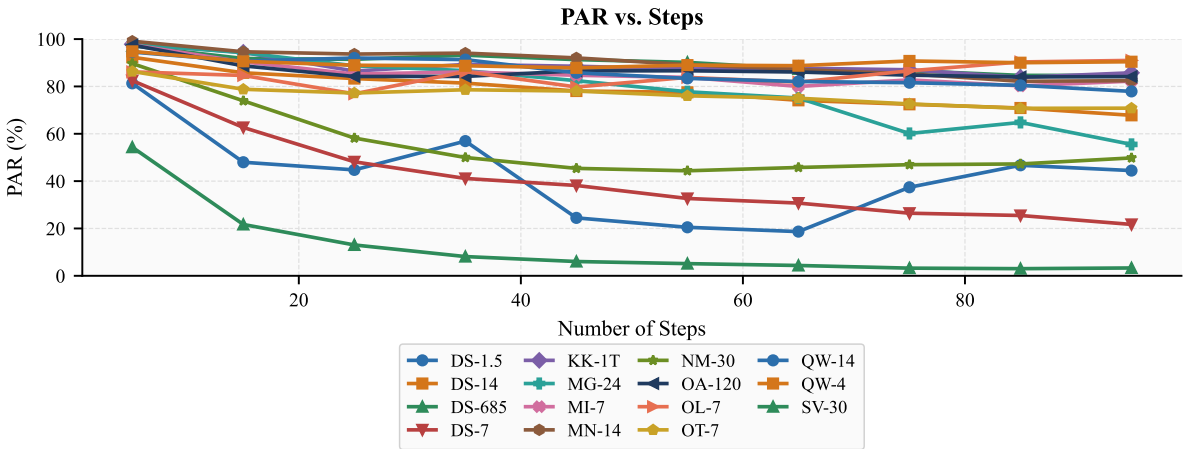


Figure 18: Coverage (non-null answer rate) across increasing step counts. While many models maintain high coverage, several exhibit declining rates with longer procedures, indicating a growing tendency to fail in producing a valid structured answer as complexity increases.

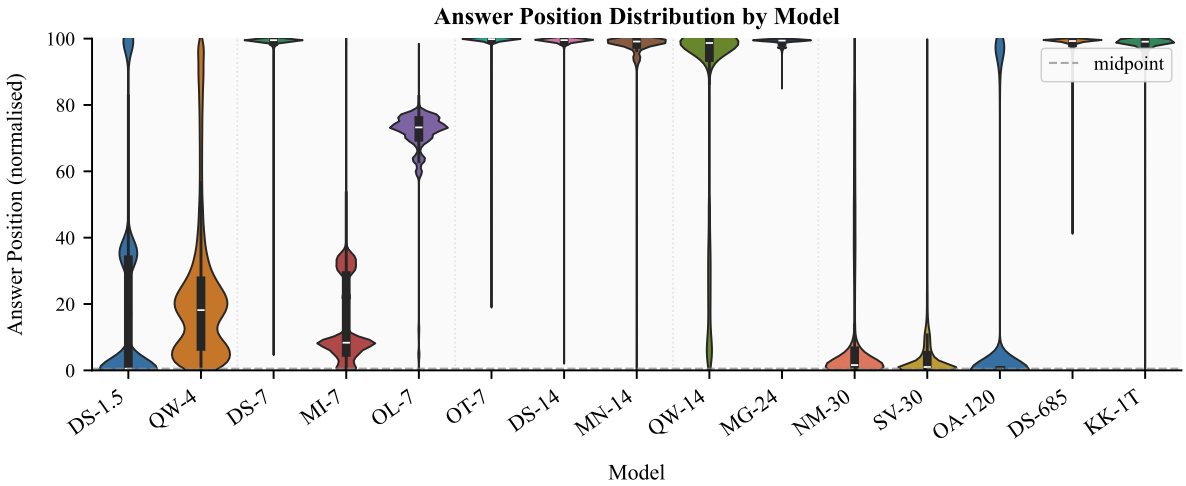


Figure 19: Distribution of the normalized position of the first generated answer across models. Models vary in when they commit to an answer, with some producing early answers and others delaying until later in the output, reflecting different generation strategies rather than consistent differences in correctness.

G Results: Model wise Analysis

G.1 FAA Across Data Types

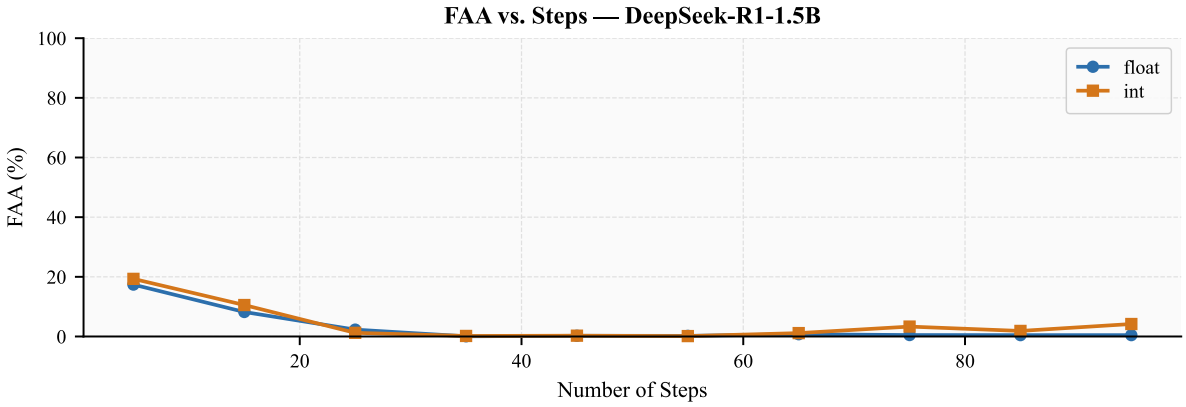


Figure 20: FAA as a function of procedure length for integer and floating-point input settings. FAA drops sharply as the number of steps increases, approaching near-zero for longer sequences, with no significant difference between integer and floating-point inputs.

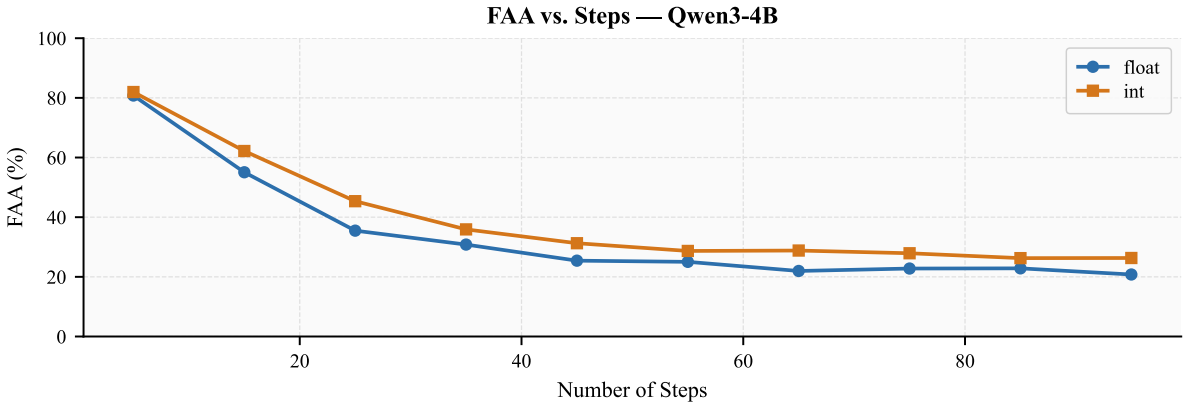


Figure 21: FAA as a function of procedure length for integer and floating-point input settings. FAA drops as the number of steps increases. We can see the FAA is slghtly better for integer inputs then floating-point inputs.

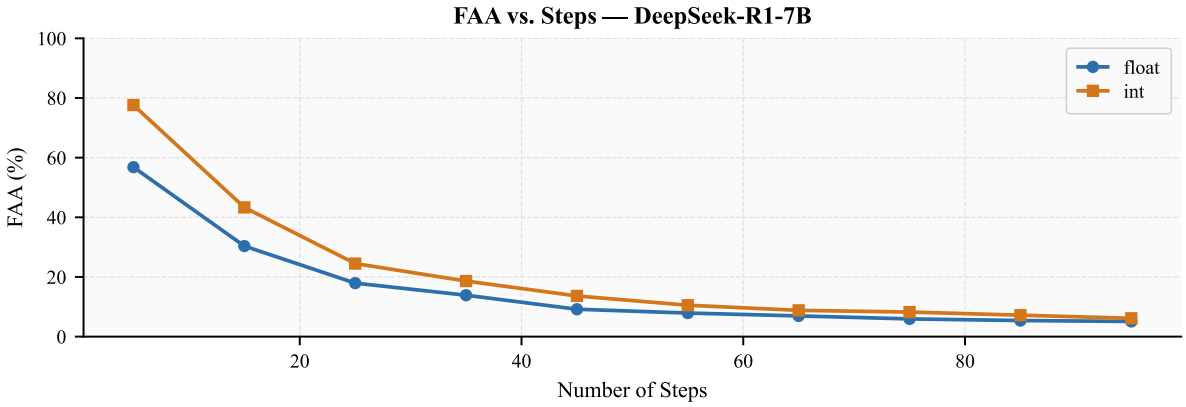


Figure 22: FAA as a function of procedure length for integer and floating-point input settings. FAA drops as the number of steps increases. We can see the FAA is slghtly better for integer inputs then floating-point inputs but as the step length increases the converge.

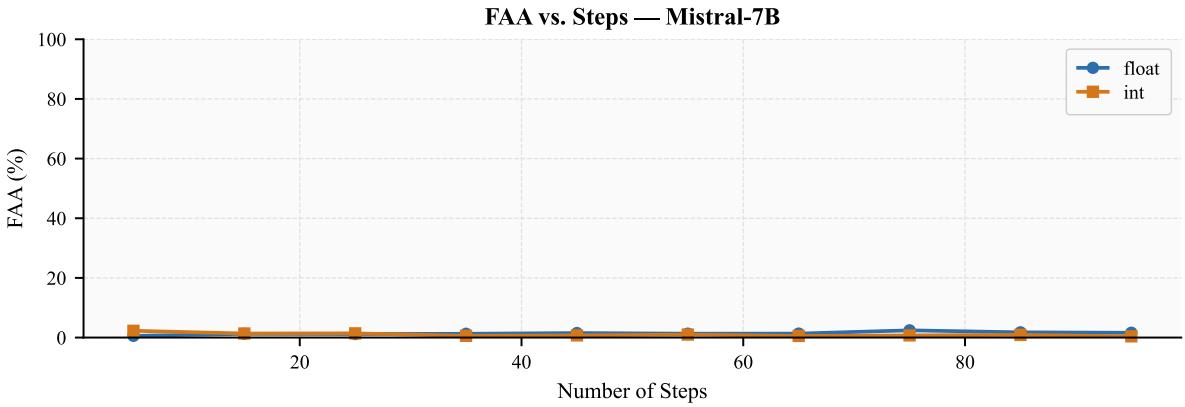


Figure 23: FAA as a function of procedure length for integer and floating-point input settings. FAA remains consistently low across all step lengths for both data types, with no meaningful differences observed between integer and floating-point inputs.

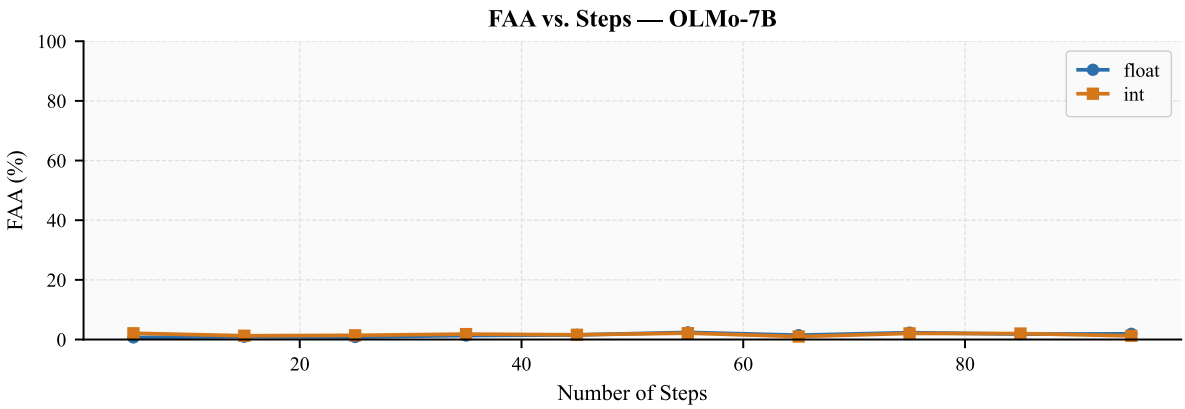


Figure 24: FAA as a function of procedure length for integer and floating-point input settings. FAA remains consistently low across all step lengths for both data types, with no meaningful differences observed between integer and floating-point inputs.

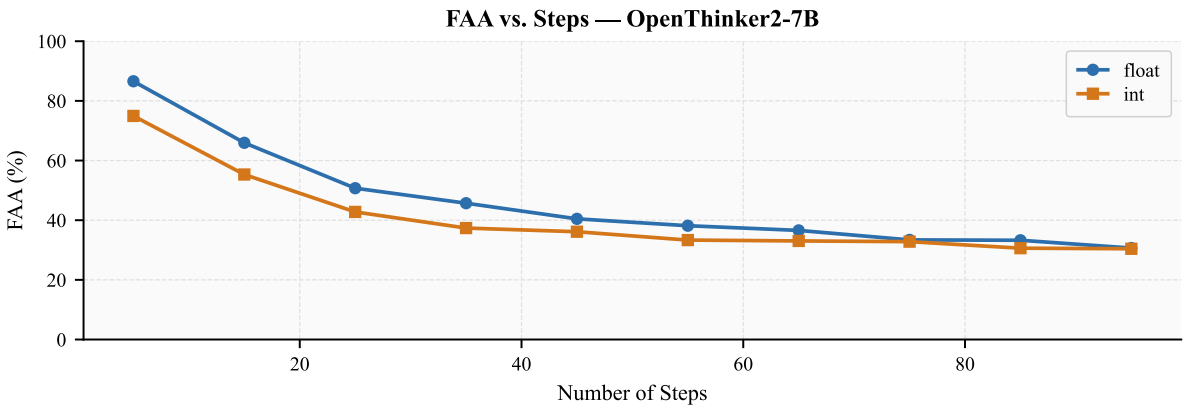


Figure 25: FAA as a function of procedure length for integer and floating-point input settings. FAA drops as the number of steps increases. We can see the FAA is slghtly better for floating-point inputs then integer inputs.

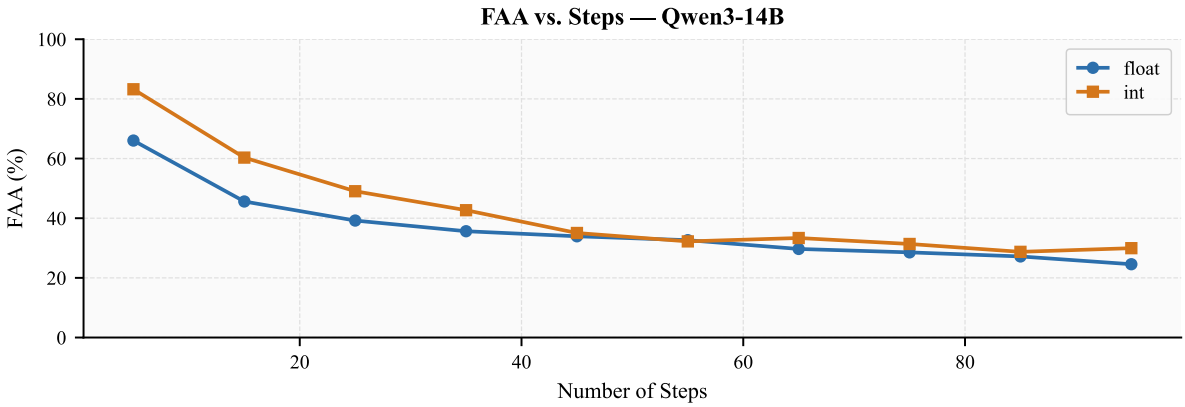


Figure 26: FAA as a function of procedure length for integer and floating-point input settings. FAA drops as the number of steps increases. We can see the FAA is slghtly better for inetger inputs then floating point inputs.

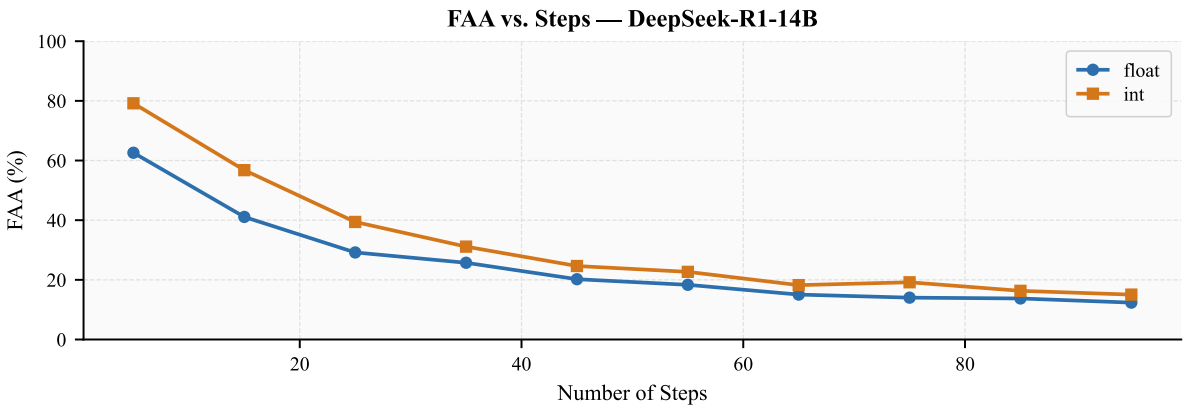


Figure 27: FAA as a function of procedure length for integer and floating-point input settings. FAA drops as the number of steps increases. We can see the FAA is slghtly better for integer inputs then floating-point inputs.

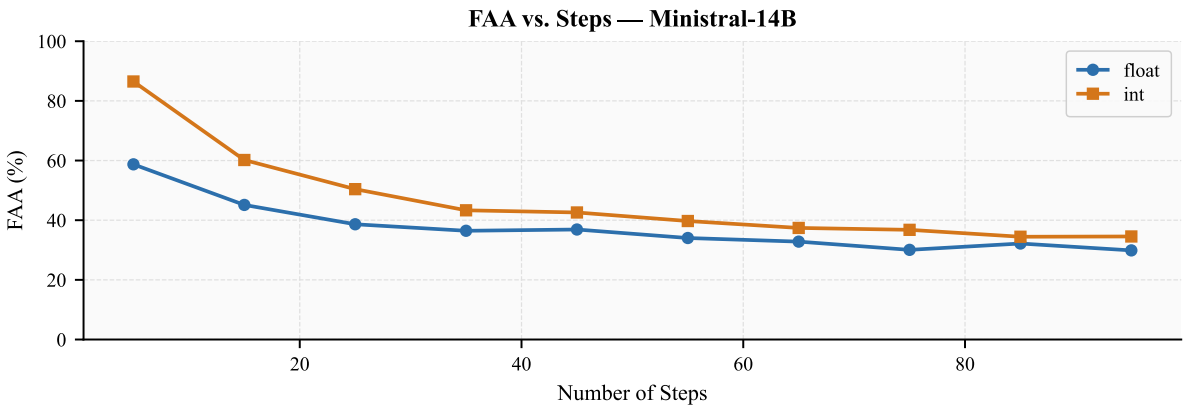


Figure 28: FAA as a function of procedure length for integer and floating-point input settings. FAA drops as the number of steps increases. We can see the FAA is slghtly better for integer inputs then floating-point inputs.

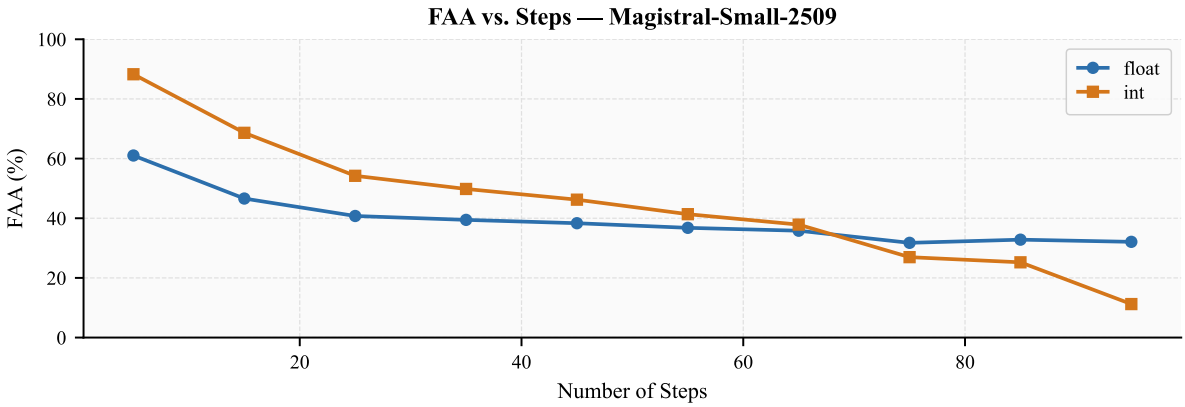


Figure 29: FAA as a function of procedure length for integer and floating-point input settings. FAA declines with increasing step length for both data types. Integer inputs achieve higher FAA at shorter step lengths but exhibit a steeper decline as the number of steps increases, eventually performing comparably to or worse than floating-point inputs at longer step lengths.

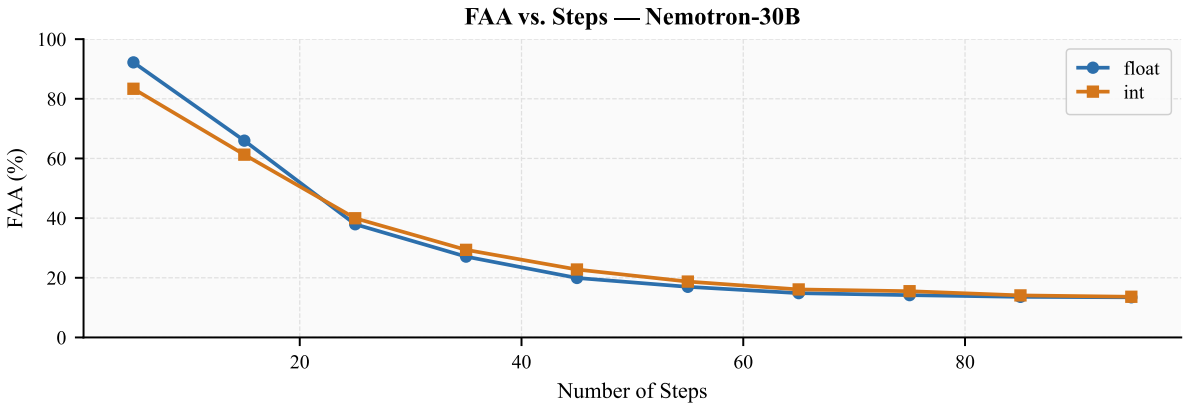


Figure 30: FAA as a function of procedure length for integer and floating-point input settings. FAA drops as the number of steps increases.

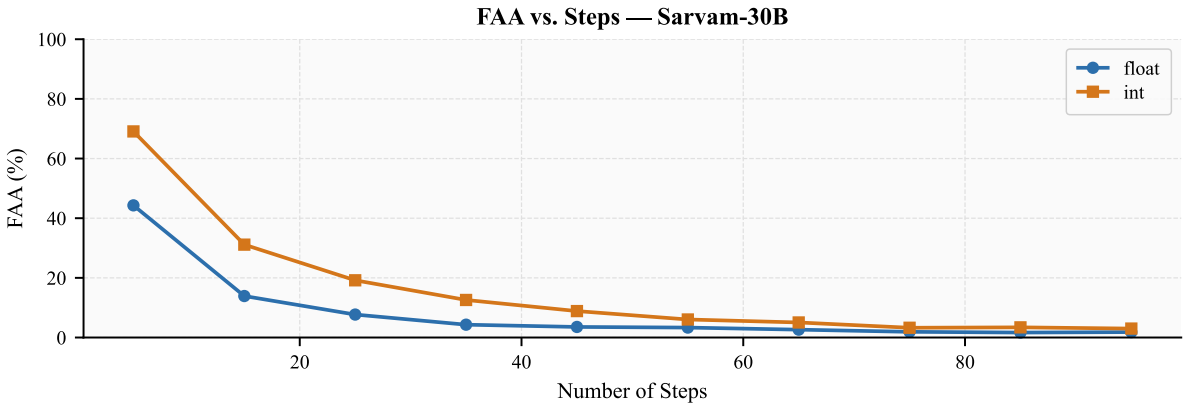


Figure 31: FAA as a function of procedure length for integer and floating-point input settings. FAA drops as the number of steps increases. We can see the FAA is slghtly better for integer inputs then floating-point inputs.

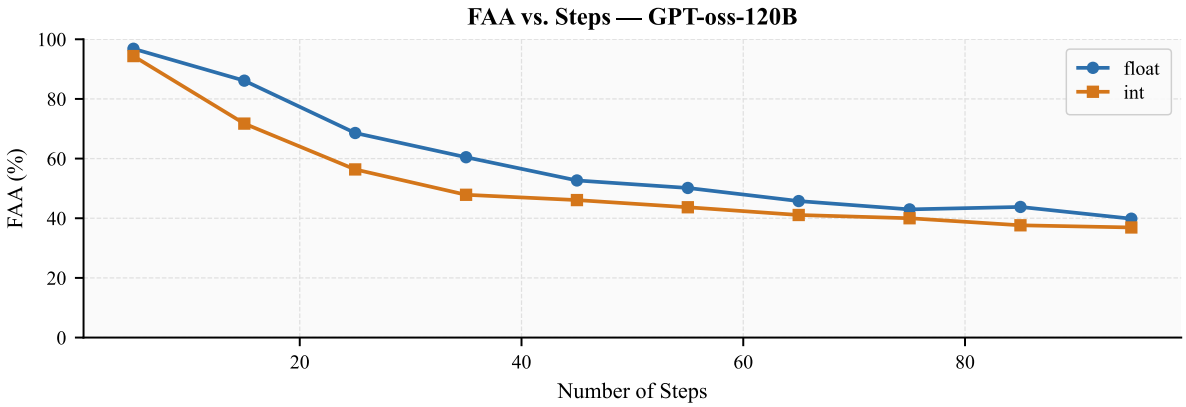


Figure 32: FAA as a function of procedure length for integer and floating-point input settings. FAA drops as the number of steps increases. We can see the FAA is slghtly better for floating-point inputs then integer inputs.

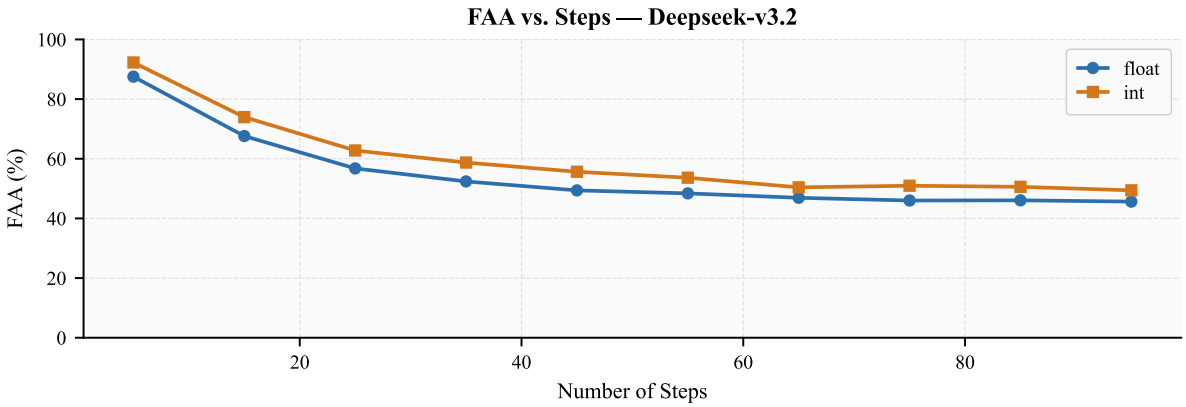


Figure 33: FAA as a function of procedure length for integer and floating-point input settings. FAA drops as the number of steps increases. We can see the FAA is slghtly better for integer inputs then floating-point inputs.

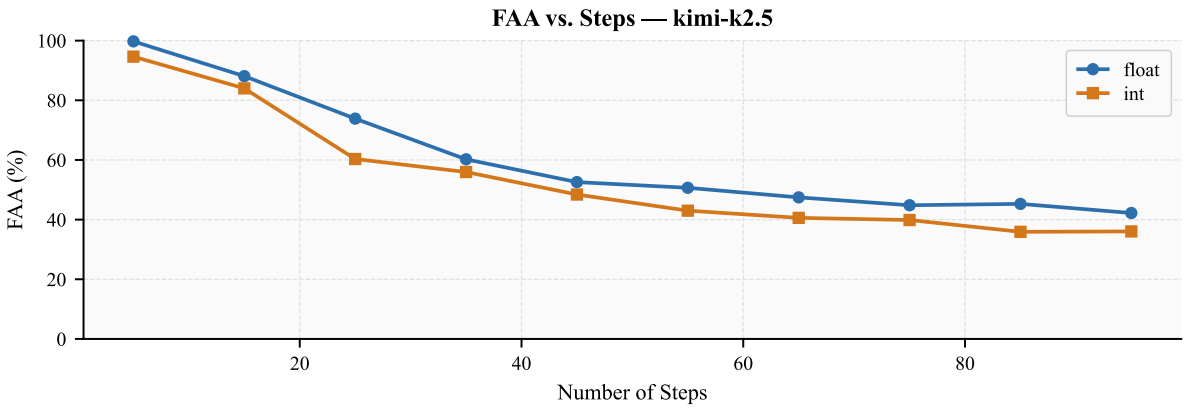


Figure 34: FAA as a function of procedure length for integer and floating-point input settings. FAA drops as the number of steps increases. We can see the FAA is slghtly better for floating-point inputs then integer inputs.

G.2 FAA Across Range of Input

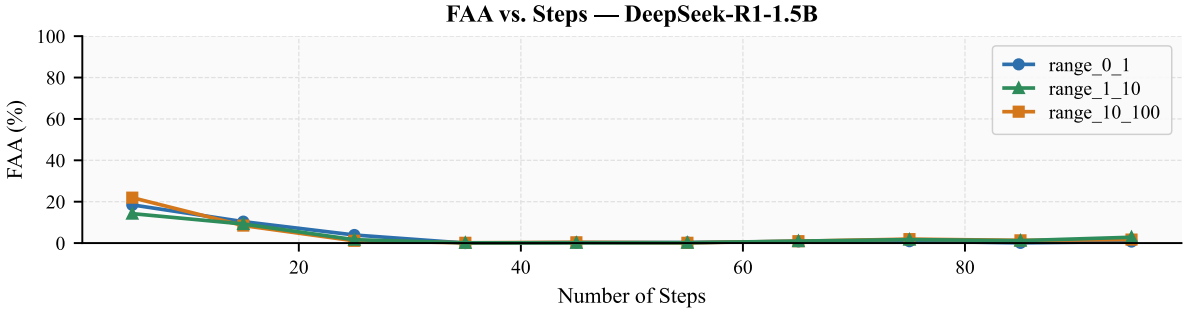


Figure 35: FAA across procedure lengths for different input ranges. Performance degrades rapidly with increasing steps across all input ranges, with minimal differences observed between ranges.

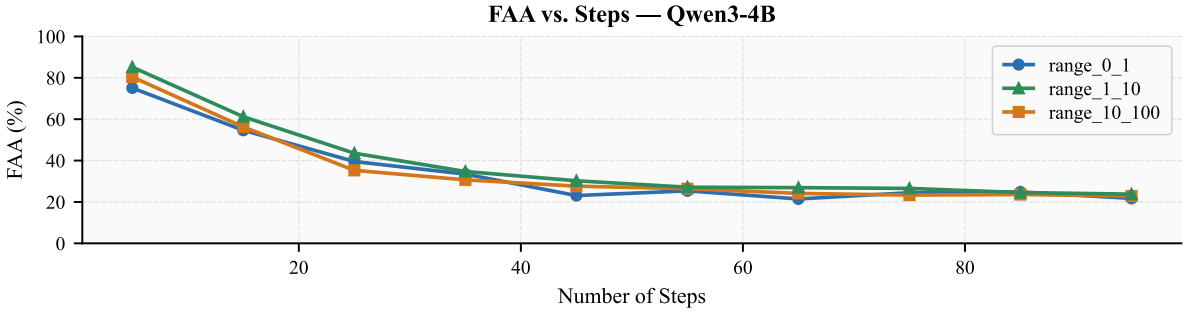


Figure 36: FAA across procedure lengths for different input ranges. Performance degrades with increasing steps across all input ranges, with minimal differences observed between ranges.

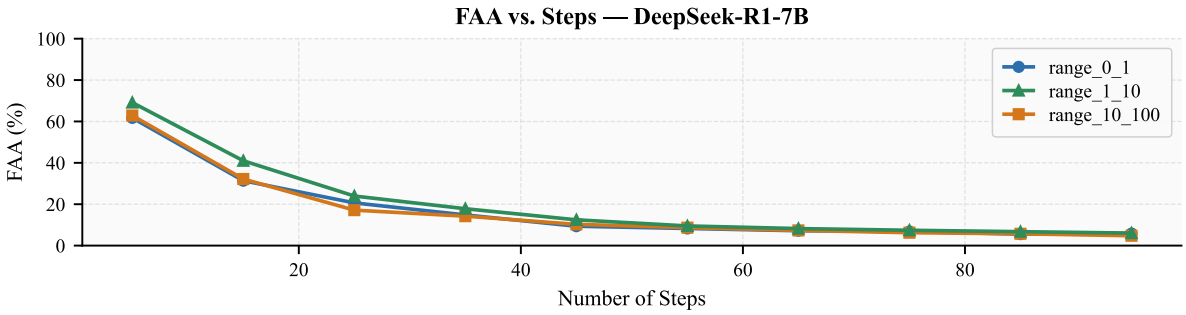


Figure 37: FAA across procedure lengths for different input ranges. Performance degrades with increasing steps across all input ranges, with minimal differences observed between ranges.

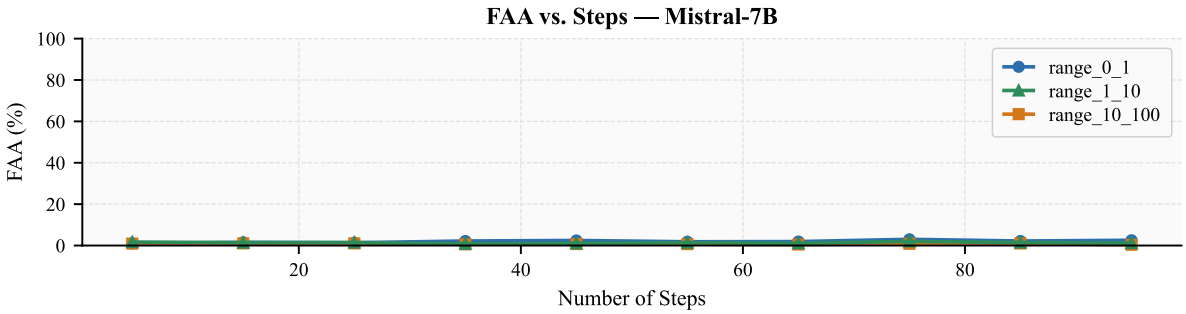


Figure 38: FAA across procedure lengths for different input ranges. Performance remains uniformly low across all input ranges and step lengths, with minimal variation between ranges.

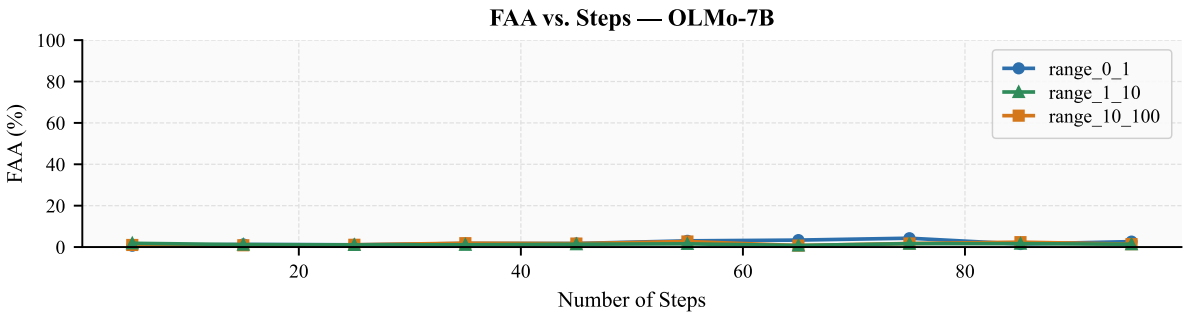


Figure 39: FAA across procedure lengths for different input ranges. Performance remains uniformly low across all input ranges and step lengths, with minimal variation between ranges.

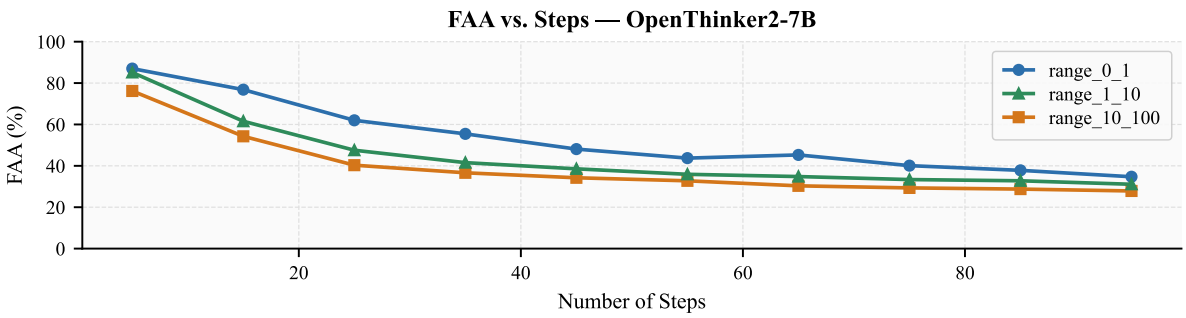


Figure 40: FAA across procedure lengths for different input ranges. Performance declines with increasing step length across all input ranges, with slightly higher FAA observed for inputs in the range [0, 1].

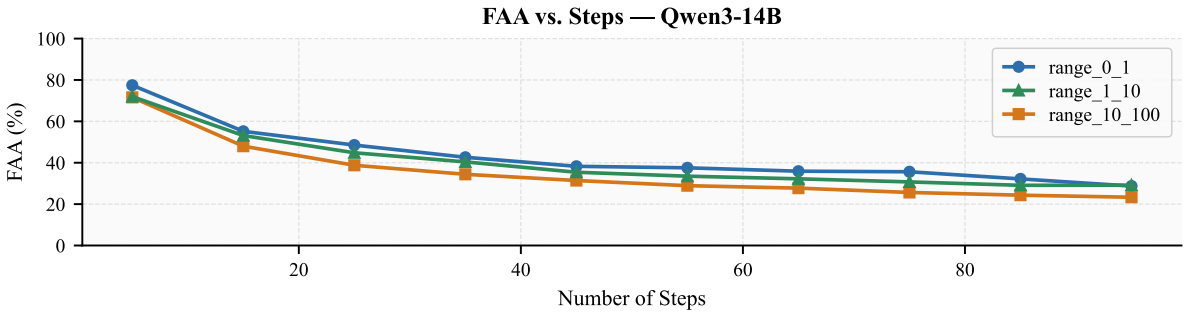


Figure 41: FAA across procedure lengths for different input ranges. Performance declines with increasing step length across all input ranges, with slightly higher FAA observed for inputs in the range [0, 1].

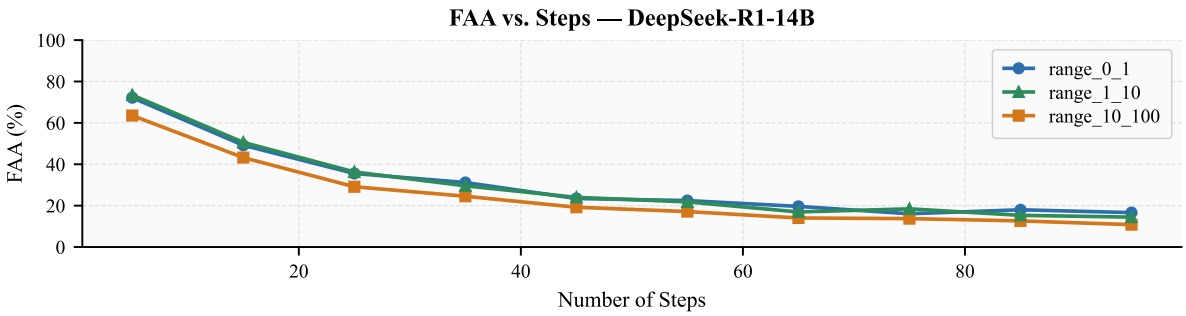


Figure 42: FAA across procedure lengths for different input ranges. Performance declines with increasing step length across all input ranges, with slightly higher FAA observed for inputs in the range [0, 1].

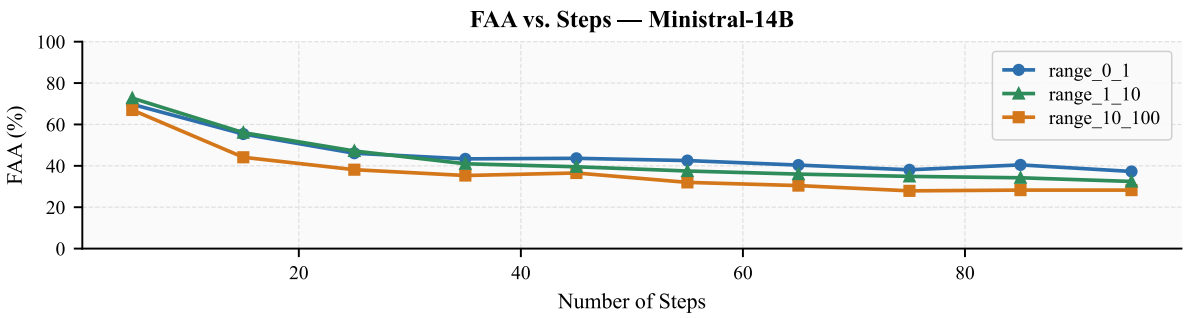


Figure 43: FAA across procedure lengths for different input ranges. Performance declines with increasing step length across all input ranges, with slightly higher FAA observed for inputs in the range [0, 1].

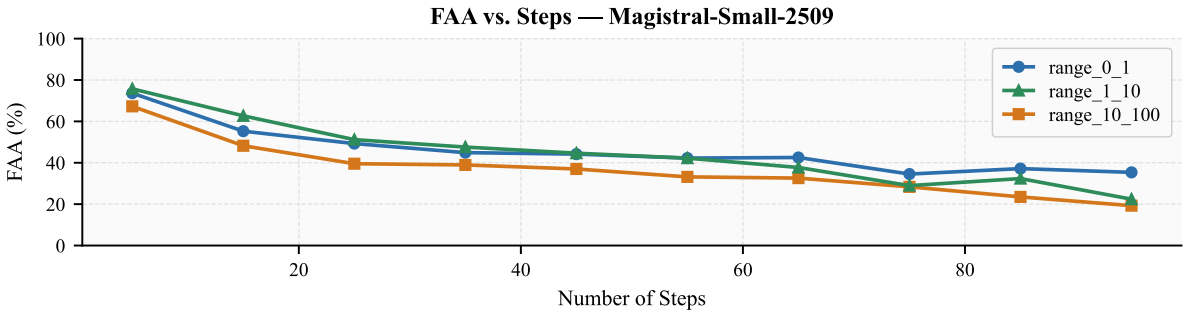


Figure 44: FAA across procedure lengths for different input ranges. Performance declines with increasing step length across all input ranges, with slightly higher FAA observed for inputs in the range [0, 1].

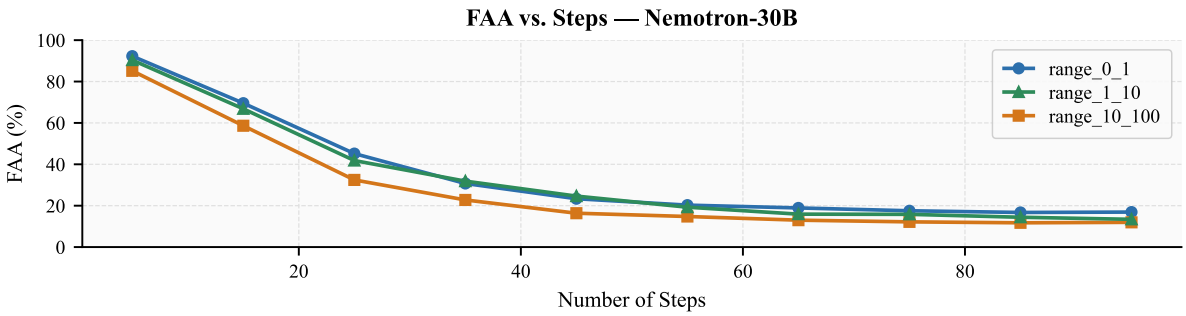


Figure 45: FAA across procedure lengths for different input ranges. Performance declines with increasing step length across all input ranges, with slightly higher FAA observed for inputs in the range [0, 1].

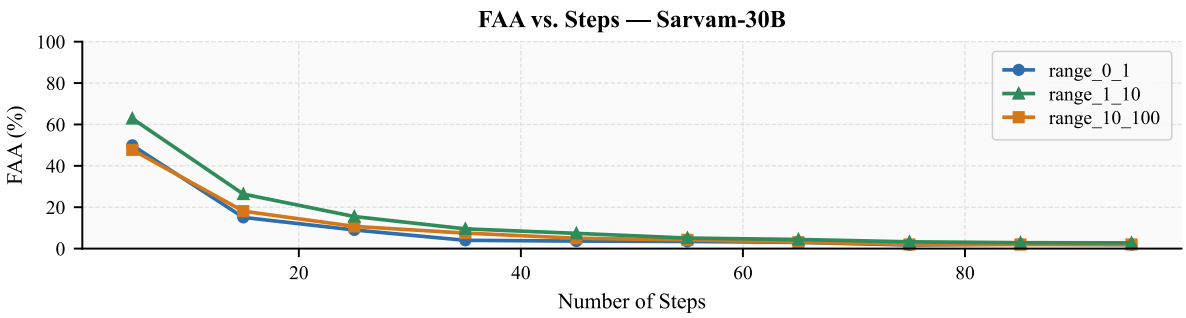


Figure 46: FAA across procedure lengths for different input ranges. Performance declines with increasing step length across all input ranges, with slightly higher FAA observed for inputs in the range [1, 10].

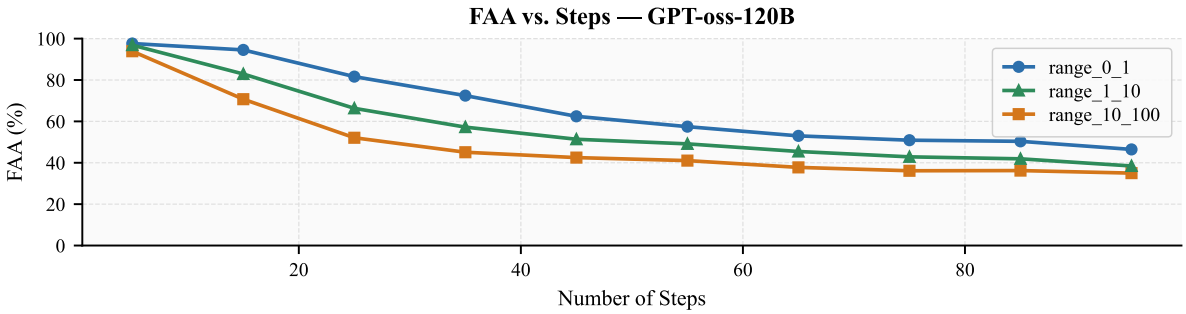


Figure 47: FAA across procedure lengths for different input ranges. Performance declines with increasing step length across all input ranges, with slightly higher FAA observed for inputs in the range [0, 1].

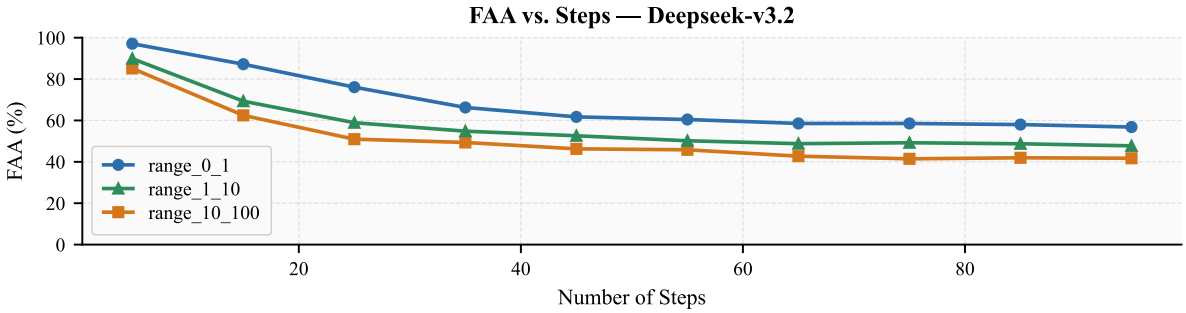


Figure 48: FAA across procedure lengths for different input ranges. Performance declines with increasing step length across all input ranges, with slightly higher FAA observed for inputs in the range [0, 1].

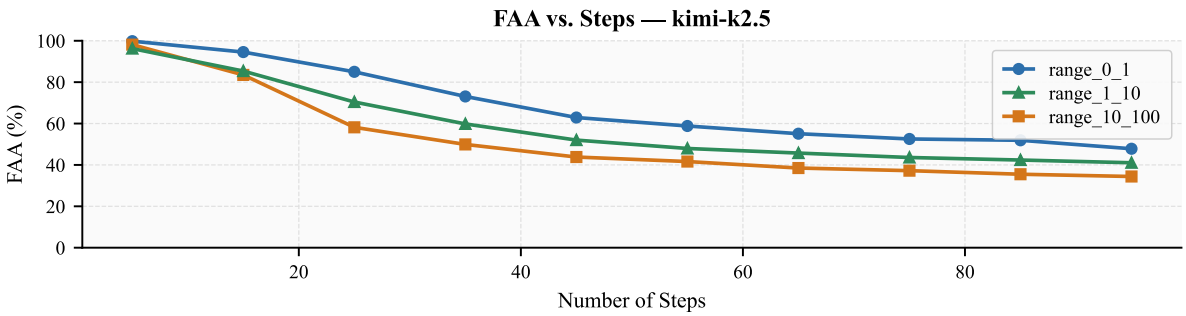


Figure 49: FAA across procedure lengths for different input ranges. Performance declines with increasing step length across all input ranges, with slightly higher FAA observed for inputs in the range [0, 1].

G.3 FAA across task variants

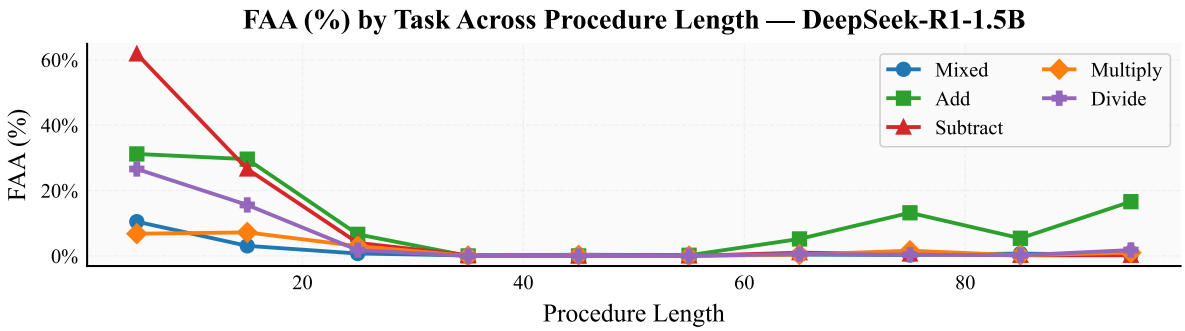


Figure 50: FAA (%) across arithmetic task variants as procedure length increases. We can se a sharp decline in FAA (%) for all Task at step 25.

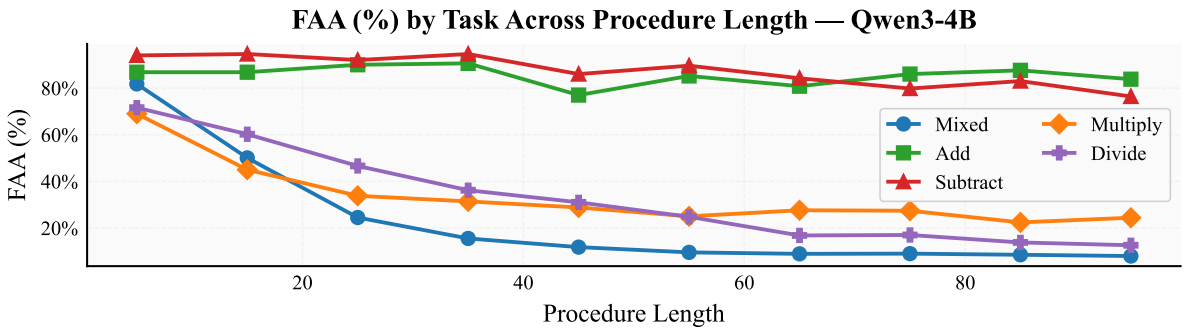


Figure 51: FAA (%) across arithmetic task variants as procedure length increases. We can see Model performed better at Addition and Subtraction while there is decline in FAA (%) for Multiplication, Division and Mixed.

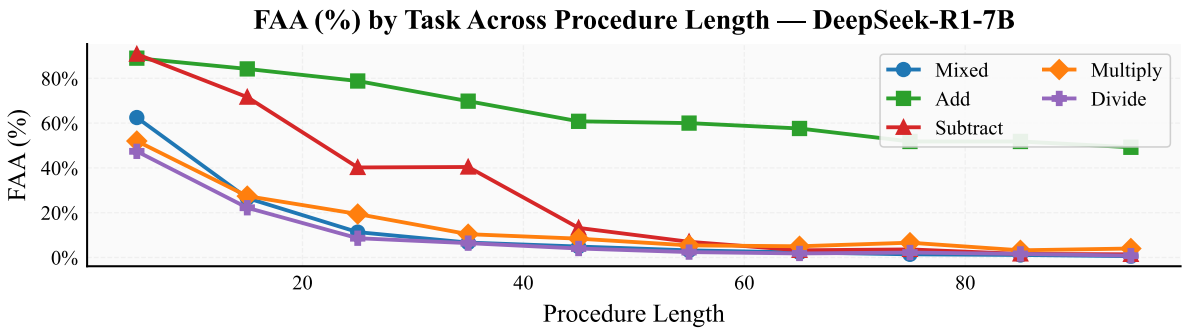


Figure 52: FAA (%) across arithmetic task variants as procedure length increases. Multiplication, Division, and Mixed tasks exhibit a steep decline in FAA, whereas Subtraction shows a more gradual decrease. Addition consistently maintains comparatively higher performance across longer procedures.

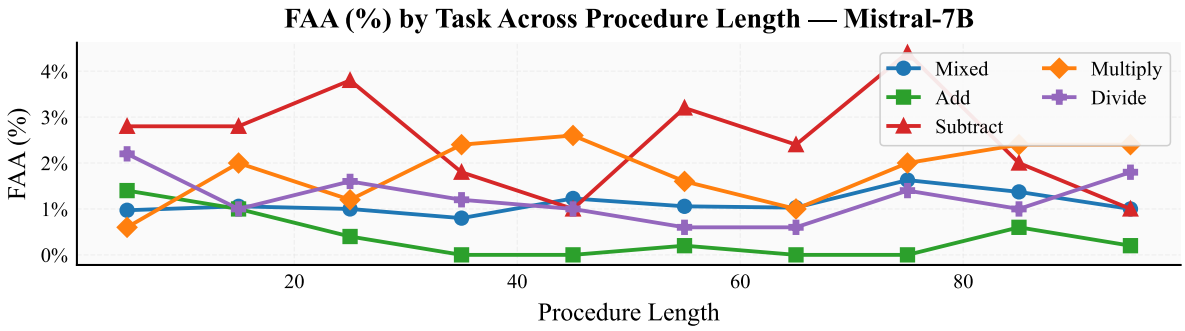


Figure 53: FAA (%) across arithmetic task variants as procedure length increases. The FAA values fluctuate considerably across procedure lengths, likely due to the overall low FAA scores, which make small performance changes appear more pronounced.

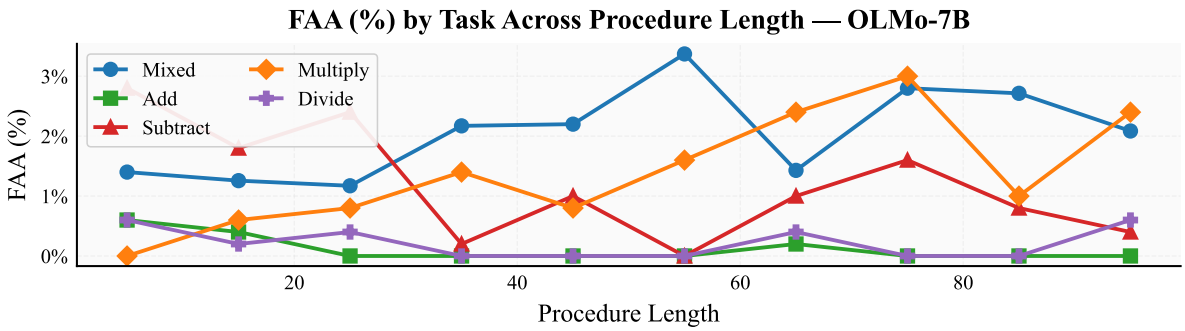


Figure 54: FAA (%) across arithmetic task variants as procedure length increases. The FAA values fluctuate considerably across procedure lengths, likely due to the overall low FAA scores, which make small performance changes appear more pronounced.

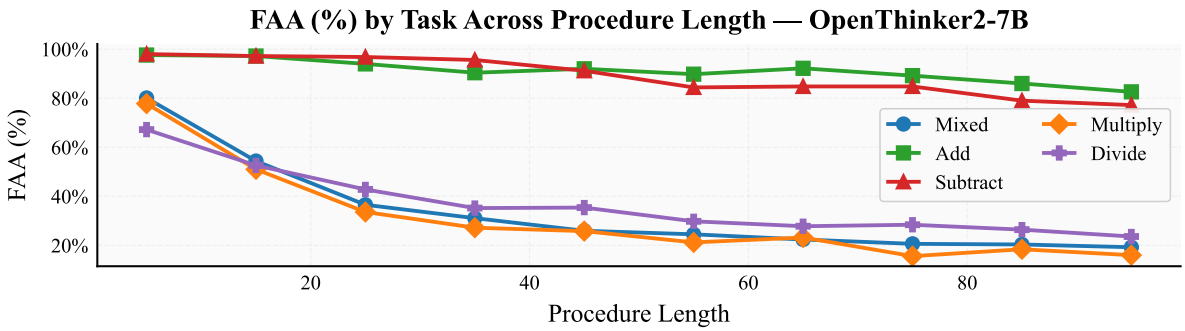


Figure 55: FAA (%) across arithmetic task variants as procedure length increases. Multiplication, Division, and Mixed tasks show a gradual decline in FAA as procedure length increases, while Addition and Subtraction maintain consistently high performance across longer procedures.

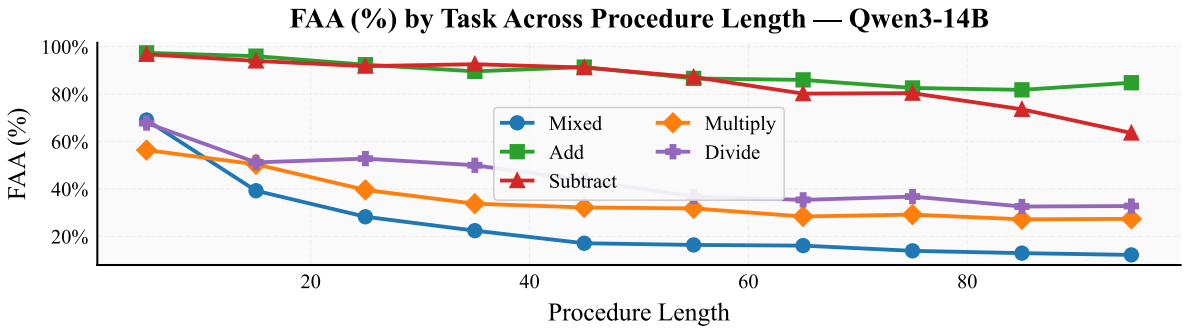


Figure 56: FAA (%) across arithmetic task variants as procedure length increases. Multiplication, Division, and Mixed tasks show a gradual decline in FAA as procedure length increases, while Addition and Subtraction maintain consistently high performance across longer procedures.

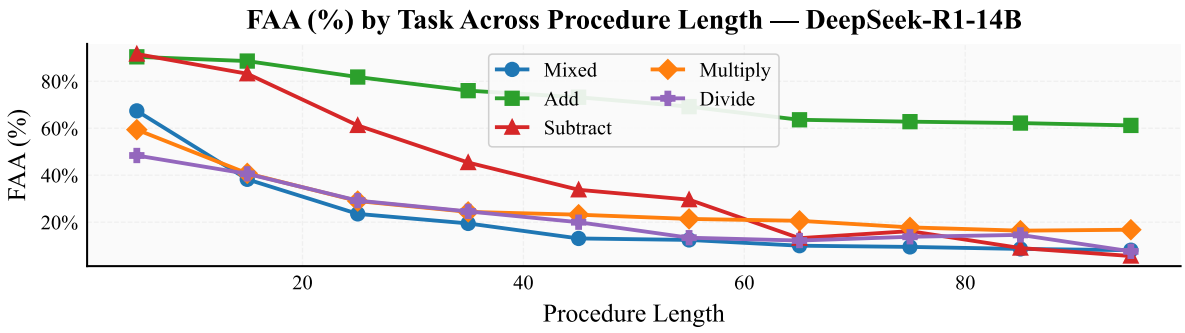


Figure 57: FAA (%) across arithmetic task variants as procedure length increases. Multiplication, Division, and Mixed tasks exhibit a steep decline in FAA, whereas Subtraction shows a more gradual decrease. Addition consistently maintains comparatively higher performance across longer procedures.

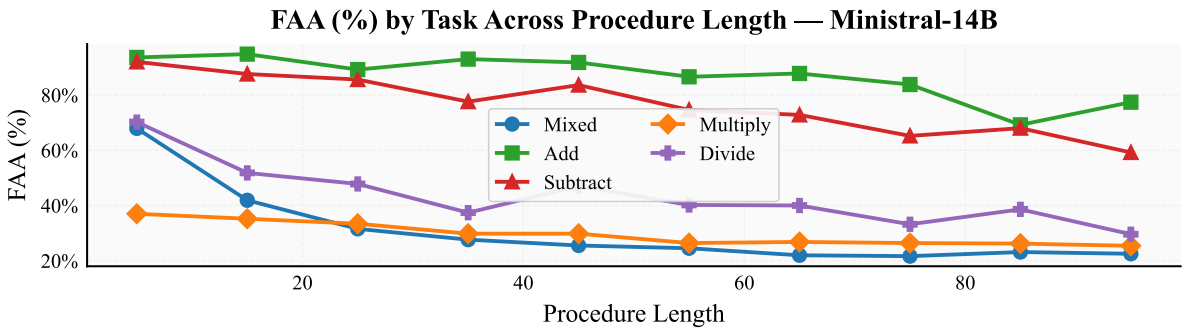


Figure 58: FAA (%) across arithmetic task variants as procedure length increases. Multiplication, Division, and Mixed tasks show a gradual decline in FAA as procedure length increases, while Addition and Subtraction maintain consistently high performance across longer procedures.

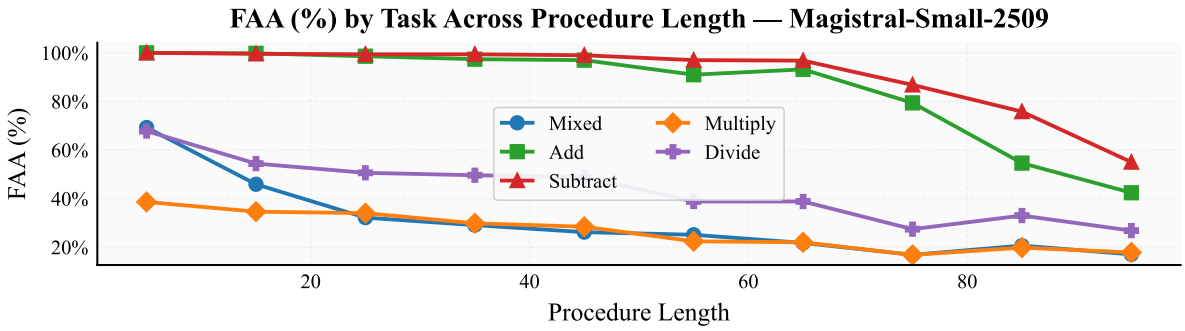


Figure 59: FAA (%) across arithmetic task variants for Magistral-Small-2509. Multiplication, Division, and Mixed tasks exhibit a gradual decline in FAA as procedure length increases, whereas Addition and Subtraction maintain consistently high performance up to around procedure length 75, after which a noticeable decline is observed.

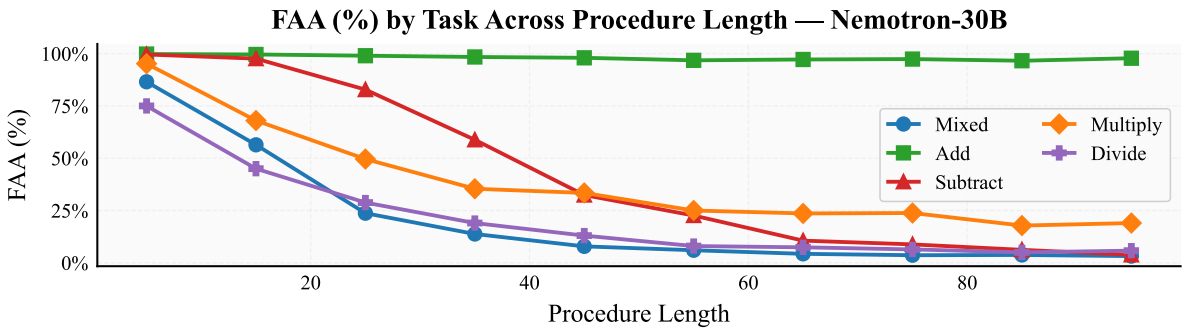


Figure 60: FAA (%) across arithmetic task variants as procedure length increases. Multiplication, Division, and Mixed tasks exhibit a steep decline in FAA, whereas Subtraction shows a more gradual decrease. Addition consistently maintains comparatively higher performance across longer procedures.

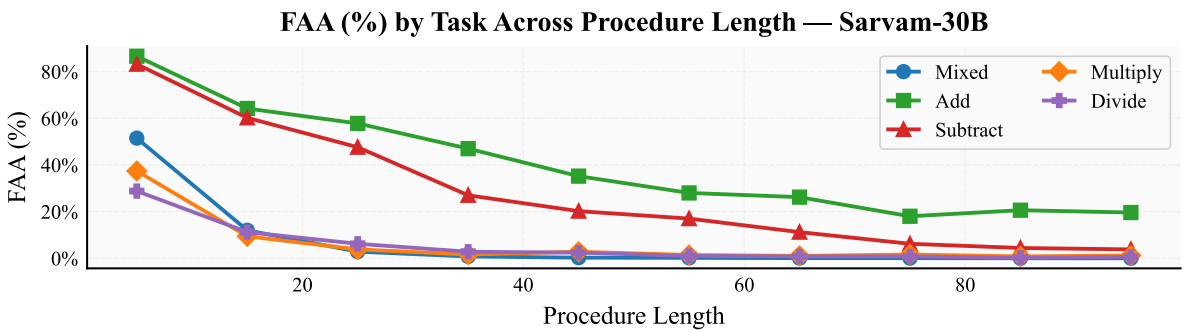


Figure 61: FAA (%) across arithmetic task variants for Sarvam-30B. Multiplication, Division, and Mixed tasks exhibit a sharp decline in FAA with increasing procedure length, reaching near-zero performance at longer lengths. Addition and Subtraction also decline steadily, though they retain comparatively higher FAA across longer procedures.

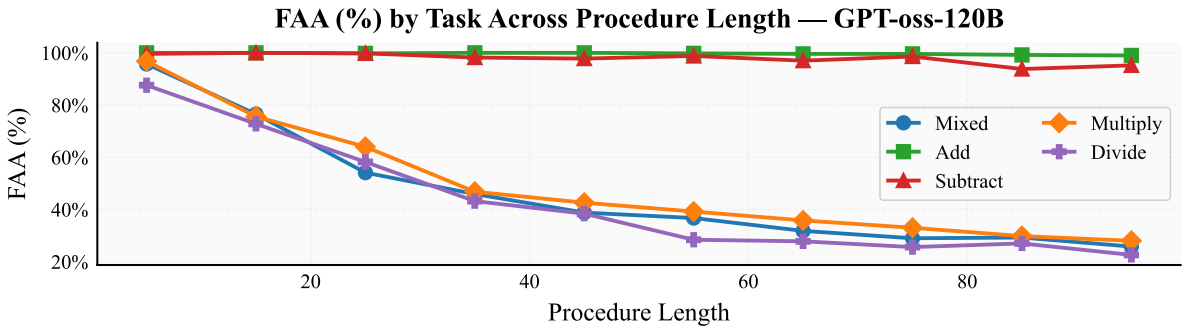


Figure 62: FAA (%) across arithmetic task variants for Magistral-Small-2509. Multiplication, Division, and Mixed tasks exhibit a gradual decline in FAA as procedure length increases, whereas Addition and Subtraction maintain consistently high performance.

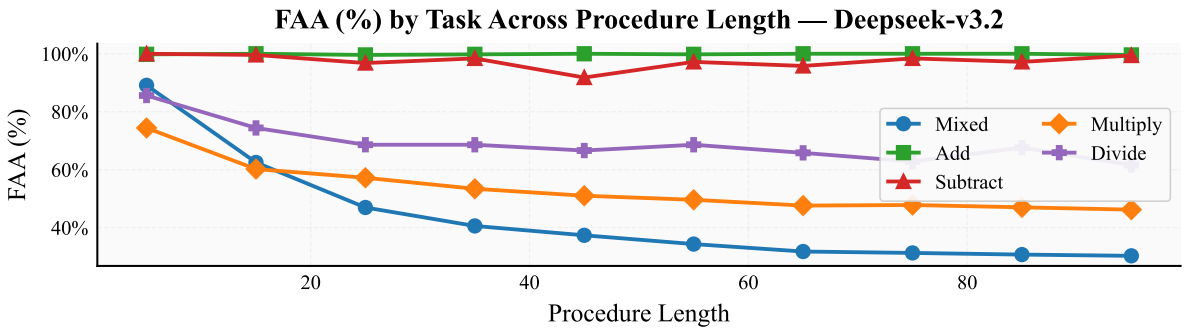


Figure 63: FAA (%) across arithmetic task variants for Magistral-Small-2509. Multiplication, Division, and Mixed tasks exhibit a gradual decline in FAA as procedure length increases, whereas Addition and Subtraction maintain consistently high performance.

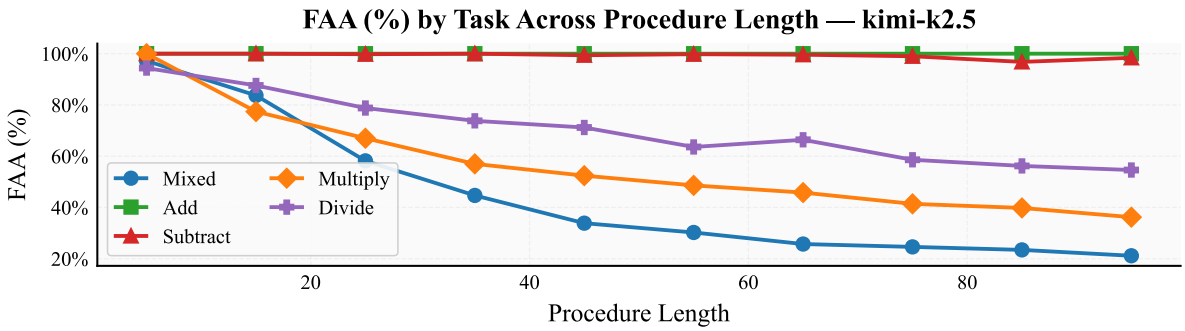


Figure 64: FAA (%) across arithmetic task variants for Magistral-Small-2509. Multiplication, Division, and Mixed tasks exhibit a gradual decline in FAA as procedure length increases, whereas Addition and Subtraction maintain consistently high performance.

G.4 Median expected output across data types.

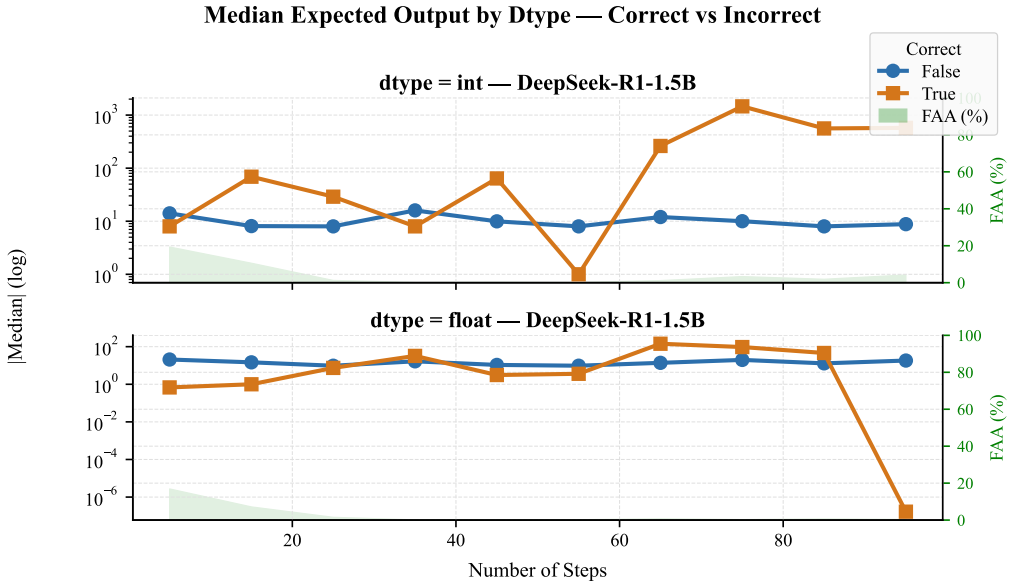


Figure 65: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). The distributions of output magnitudes for correct and incorrect predictions largely overlap across all step lengths and both data types, indicating no clear relationship between output magnitude and prediction correctness.

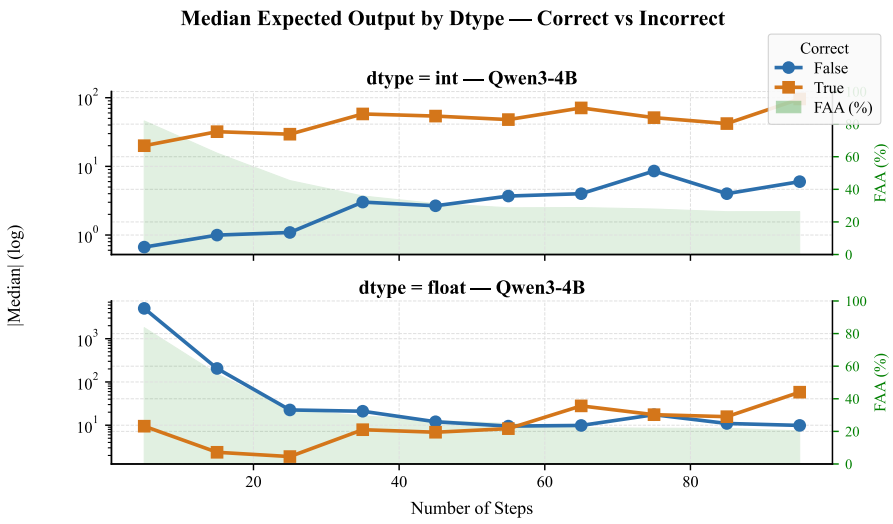


Figure 66: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For floating-point inputs, the output magnitudes for correct and incorrect predictions largely overlap across all step lengths. In contrast, for integer inputs, a more consistent separation between correct and incorrect predictions is observed, indicating a clearer relationship between output magnitude and prediction correctness.

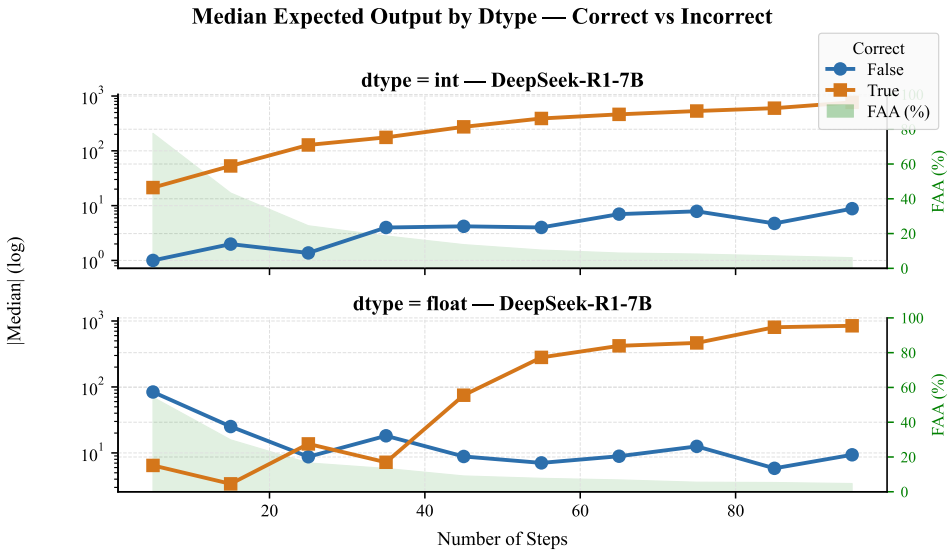


Figure 67: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For integer inputs, a consistent separation between correct and incorrect predictions is observed, with correct predictions generally associated with higher output magnitudes. For floating-point inputs, the distributions initially overlap but become increasingly separated at larger step counts, exhibiting a trend similar to integer inputs and suggesting a relationship between output magnitude and prediction correctness.

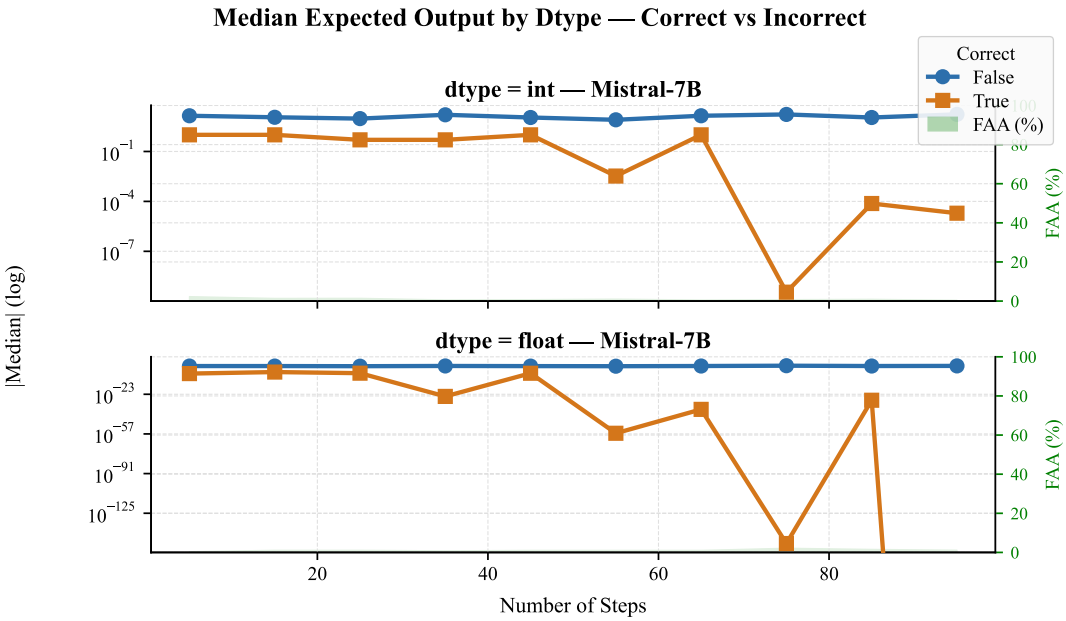


Figure 68: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For both data types, model performance tends to degrade for higher-magnitude expected outputs.

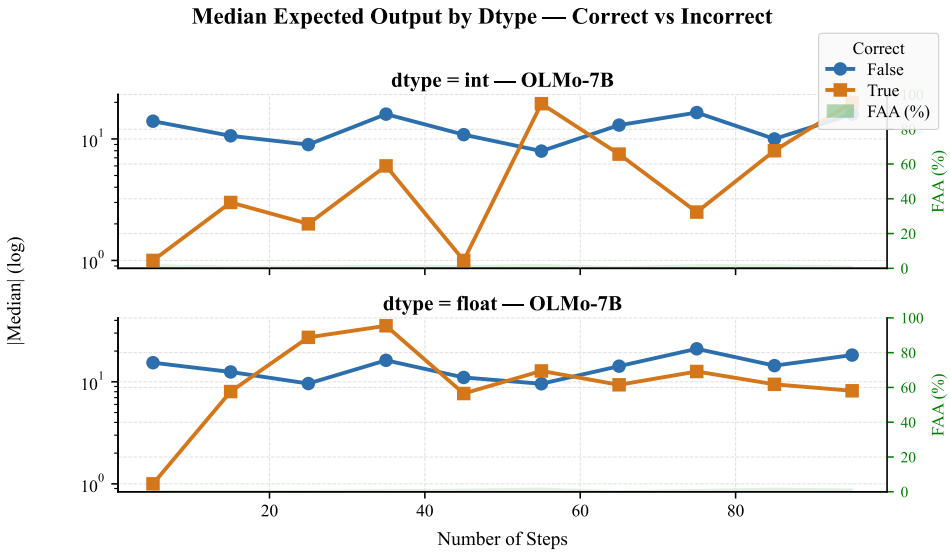


Figure 69: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For integer inputs, a separation between the output magnitudes of correct and incorrect predictions is observed, but this separation gradually diminishes as the number of steps increases. In contrast, for floating-point inputs, the distributions largely overlap across all step lengths.

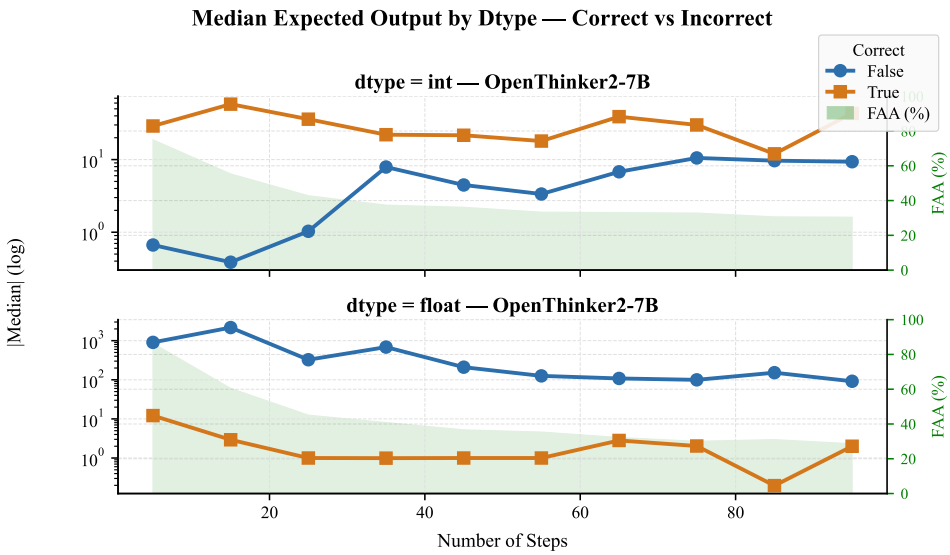


Figure 70: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For both data types, a consistent separation between correct and incorrect predictions is observed, but with opposite trends. For floating-point inputs, correct predictions are associated with lower output magnitudes, whereas for integer inputs, they are associated with higher output magnitudes.

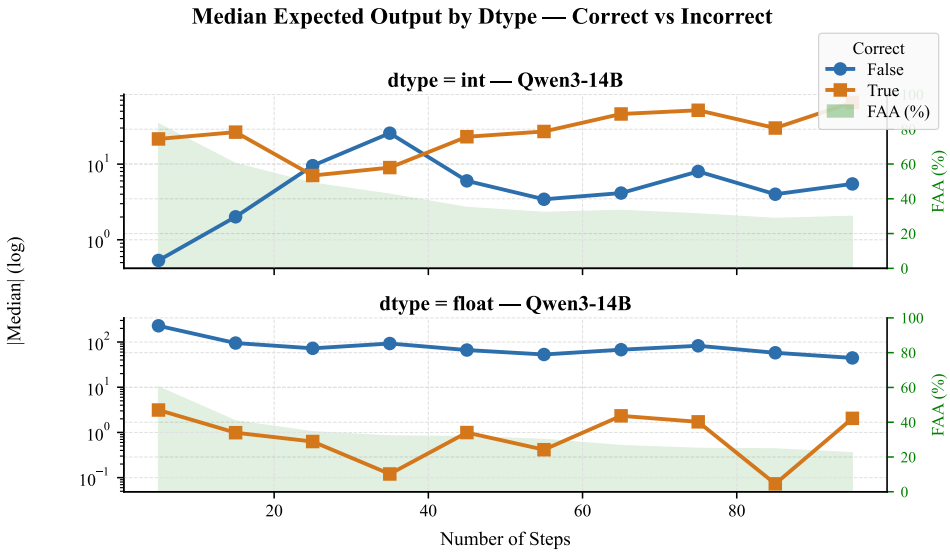


Figure 71: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For both data types, a consistent separation between correct and incorrect predictions is observed, but with opposite trends. For floating-point inputs, correct predictions are associated with lower output magnitudes, whereas for integer inputs, they are associated with higher output magnitudes.

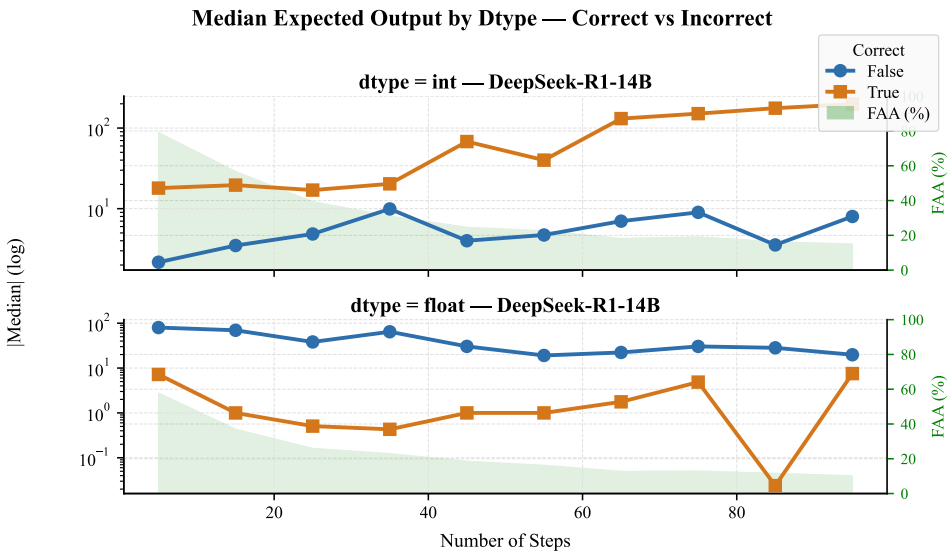


Figure 72: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For both data types, a consistent separation between correct and incorrect predictions is observed, but with opposite trends. For floating-point inputs, correct predictions are associated with lower output magnitudes, whereas for integer inputs, they are associated with higher output magnitudes.

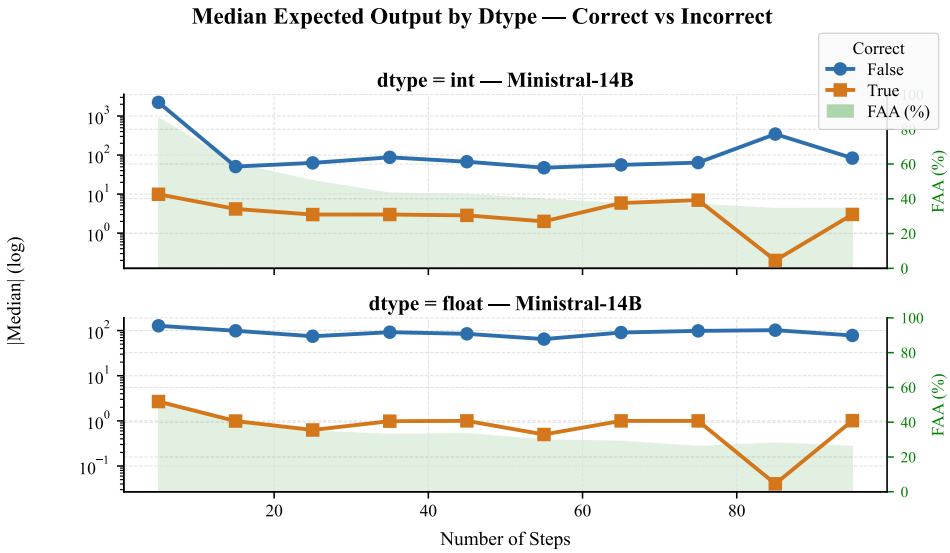


Figure 73: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For both data types, a consistent separation between correct and incorrect predictions is observed, but with opposite trends. For floating-point inputs, correct predictions are associated with lower output magnitudes, whereas for integer inputs, they are associated with higher output magnitudes.

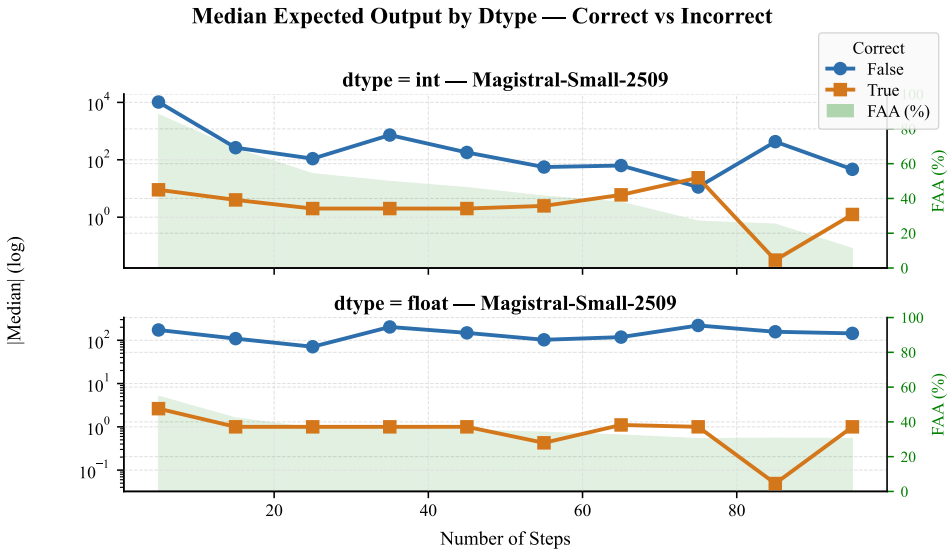


Figure 74: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For both data types, a consistent separation between correct and incorrect predictions is observed, where correct predictions are associated with lower output magnitudes in both the cases.

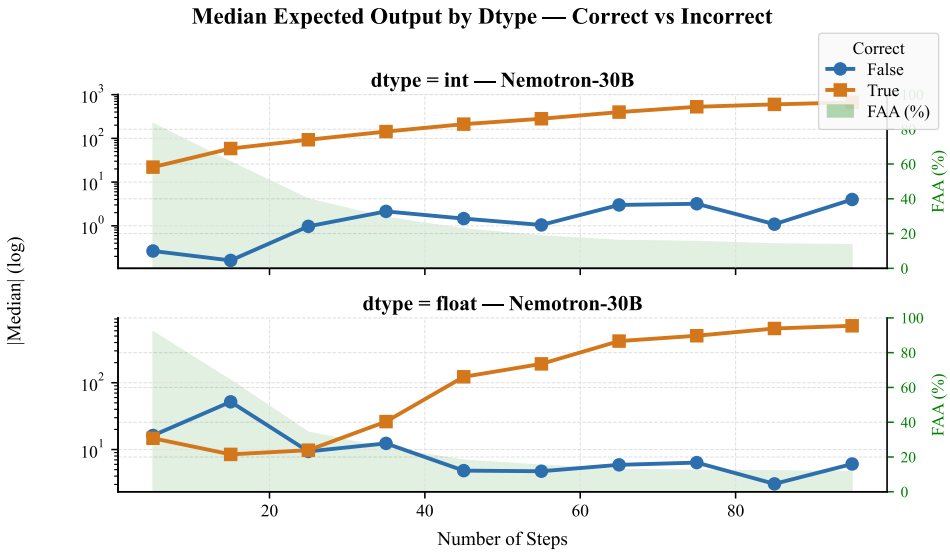


Figure 75: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For both data types, a consistent separation between correct and incorrect predictions is observed, where correct predictions are associated with lower output magnitudes.

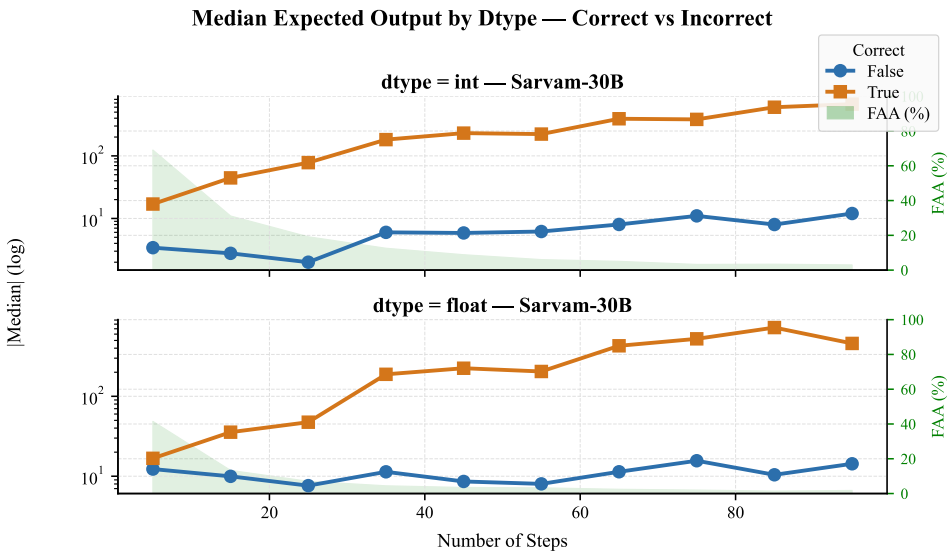


Figure 76: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For both data types, a consistent separation between correct and incorrect predictions is observed, with correct predictions generally associated with higher output magnitudes. Notably, this trend contrasts with observations from other models, where correct predictions are often associated with lower output magnitudes.

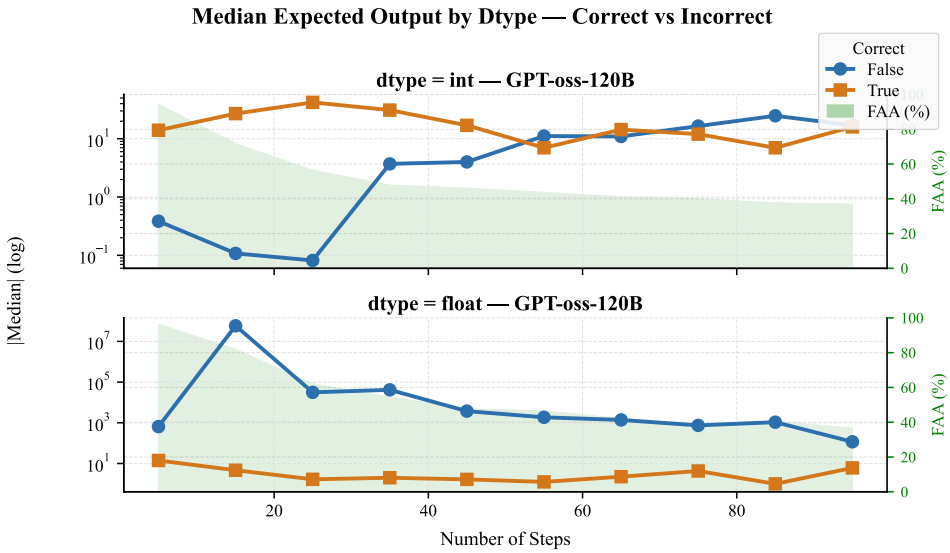


Figure 77: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For both data types, a consistent separation between correct and incorrect predictions is observed, but with opposite trends. For floating-point inputs, correct predictions are associated with lower output magnitudes, whereas for integer inputs, they are associated with higher output magnitudes but as the step increase they converged.

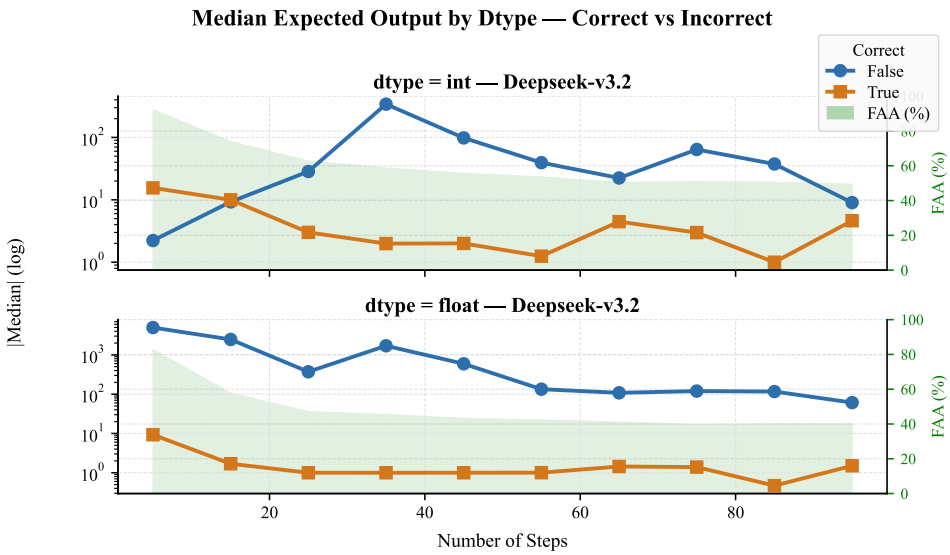


Figure 78: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For both data types, a consistent separation between correct and incorrect predictions is observed, but with opposite trends. For floating-point inputs, correct predictions are associated with lower output magnitudes, whereas for integer inputs, they are generally associated with higher output magnitudes.

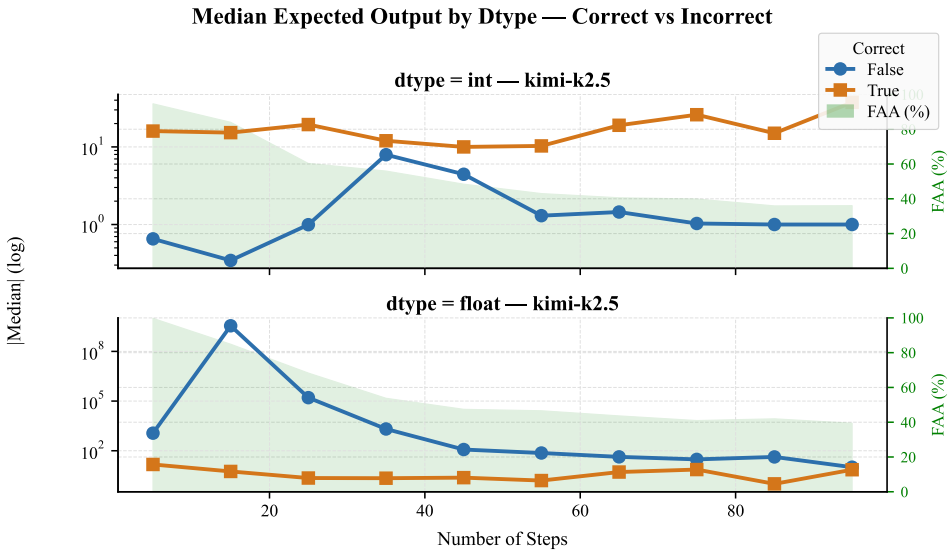


Figure 79: Median expected output across procedure lengths for integer and floating-point inputs, separated by correct and incorrect predictions (log scale). For both data types, a consistent separation between correct and incorrect predictions is observed, but with opposite trends. For floating-point inputs, correct predictions are associated with lower output magnitudes, whereas for integer inputs, they are generally associated with higher output magnitudes.

G.5 Median expected output across input ranges

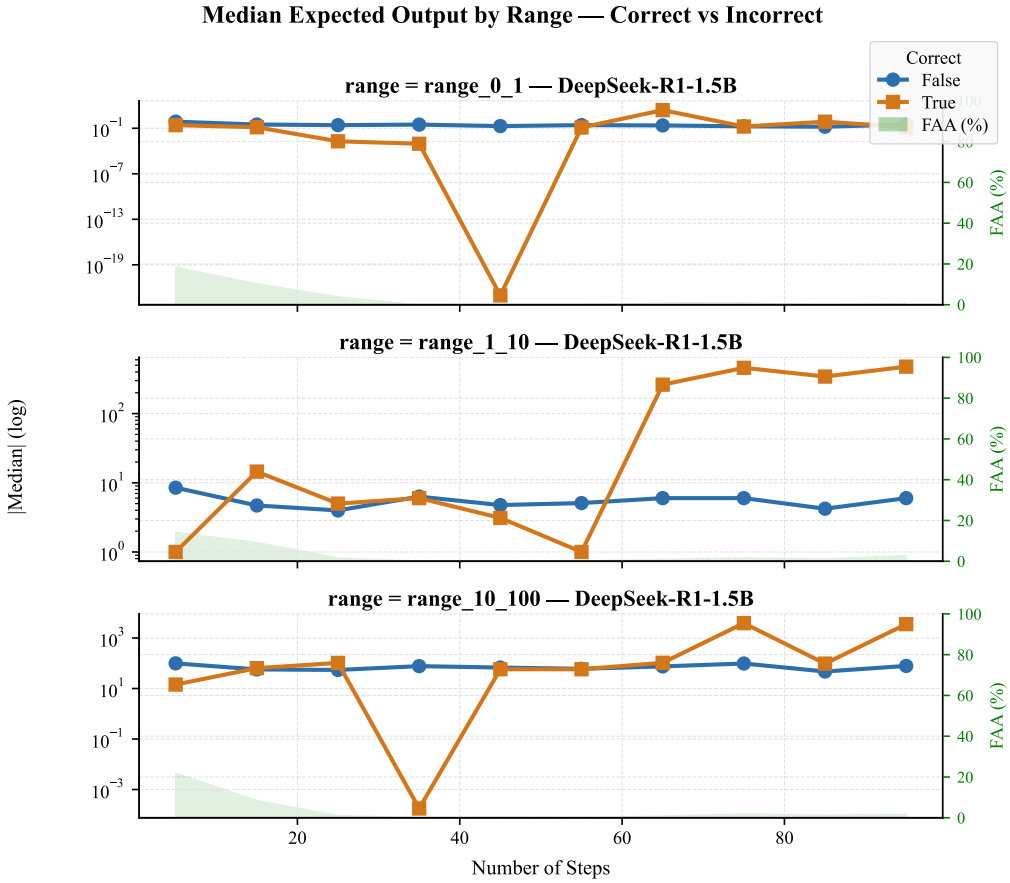


Figure 80: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). The magnitudes of expected outputs for correct and incorrect predictions exhibit substantial overlap across all ranges and step lengths, suggesting that output magnitude alone does not distinguish correct from incorrect predictions.

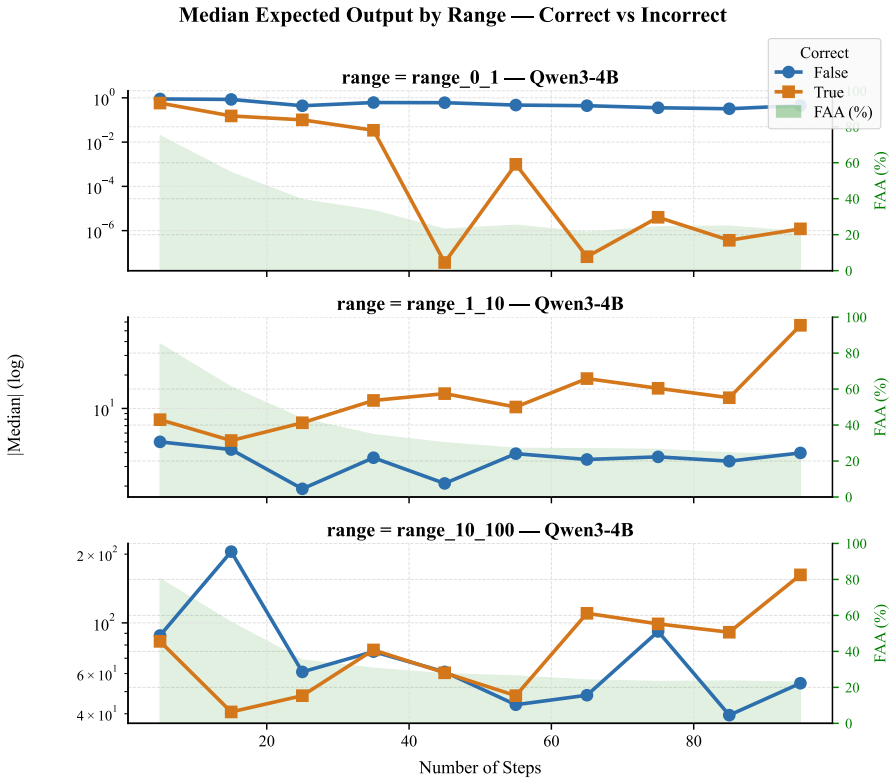


Figure 81: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). For ranges [0,1] and [1,10], a noticeable separation in output magnitudes between correct and incorrect predictions is observed, with opposite trends: in [0,1], correct predictions tend to correspond to lower output magnitudes, whereas in [1,10], they tend to correspond to higher magnitudes. In contrast, for the range [10,100], the distributions largely overlap, indicating no clear distinction based on output magnitude.

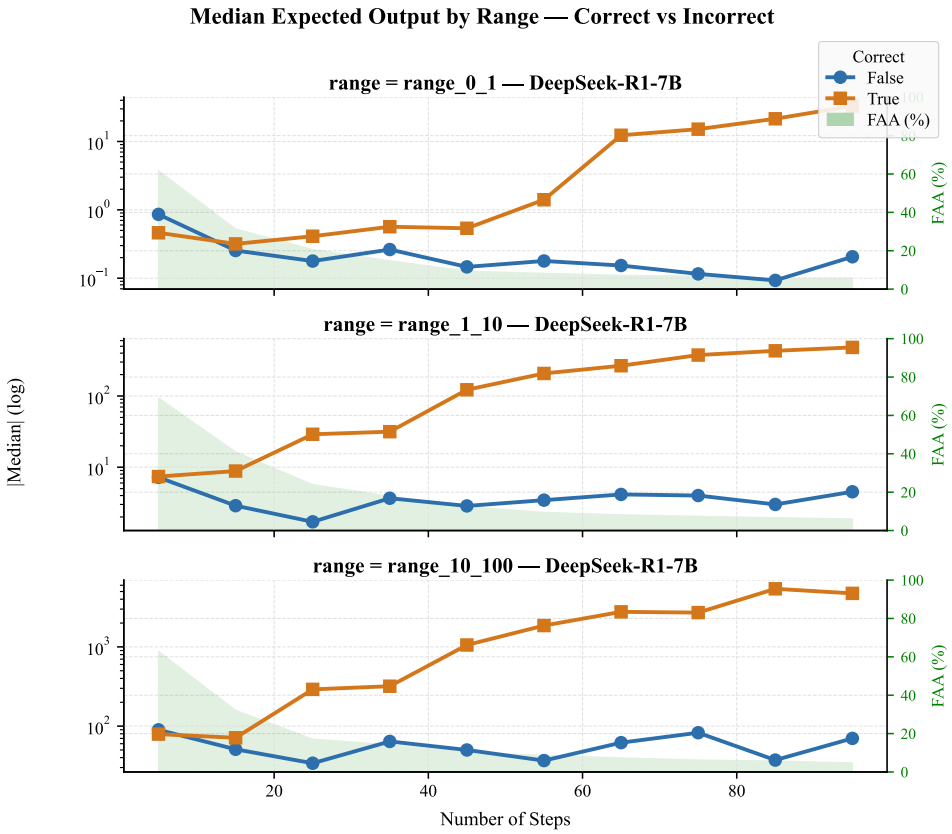


Figure 82: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). Across all ranges, a noticeable separation in output magnitudes between correct and incorrect predictions is observed, with correct predictions generally associated with higher magnitudes. This trend contrasts with the hypothesis that model performance degrades for higher-magnitude expected outputs.

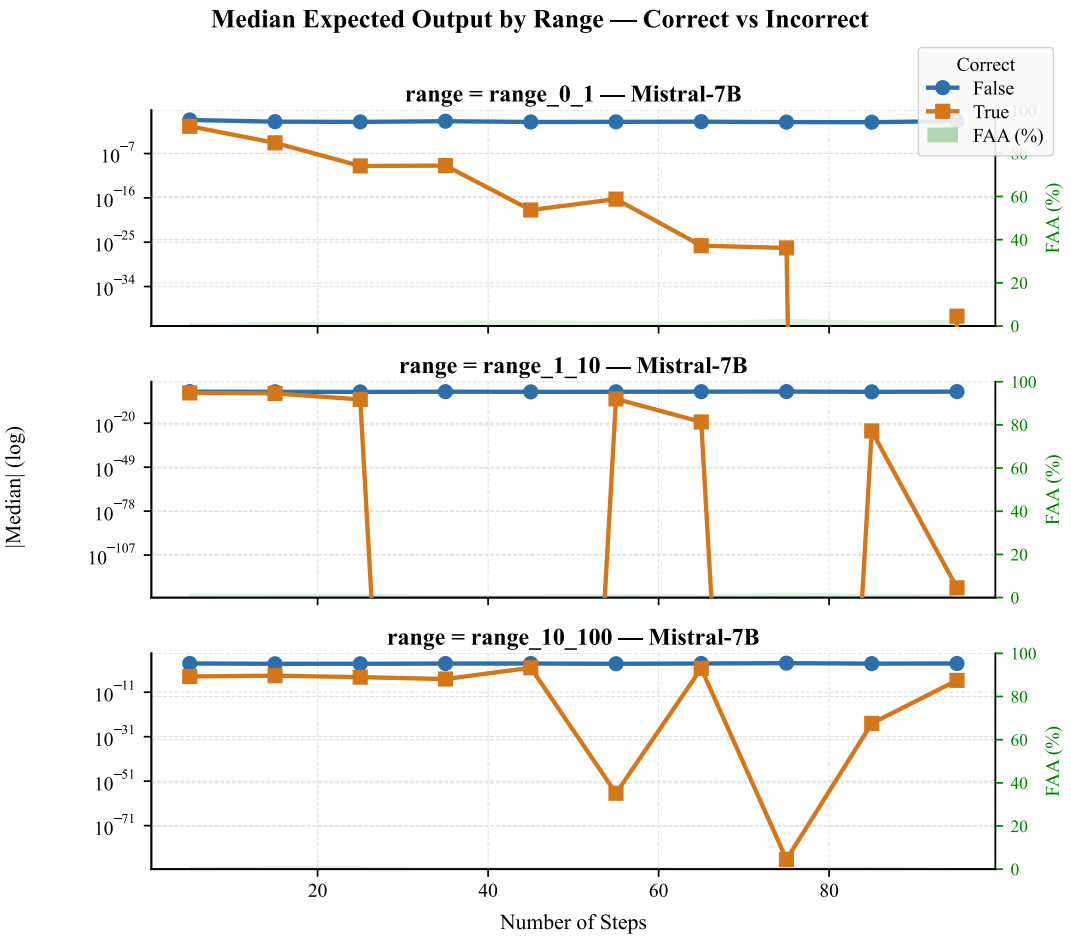


Figure 83: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). Across all three ranges, model performance tends to degrade for higher-magnitude expected outputs, suggesting that output magnitude may contribute to the observed performance decline.

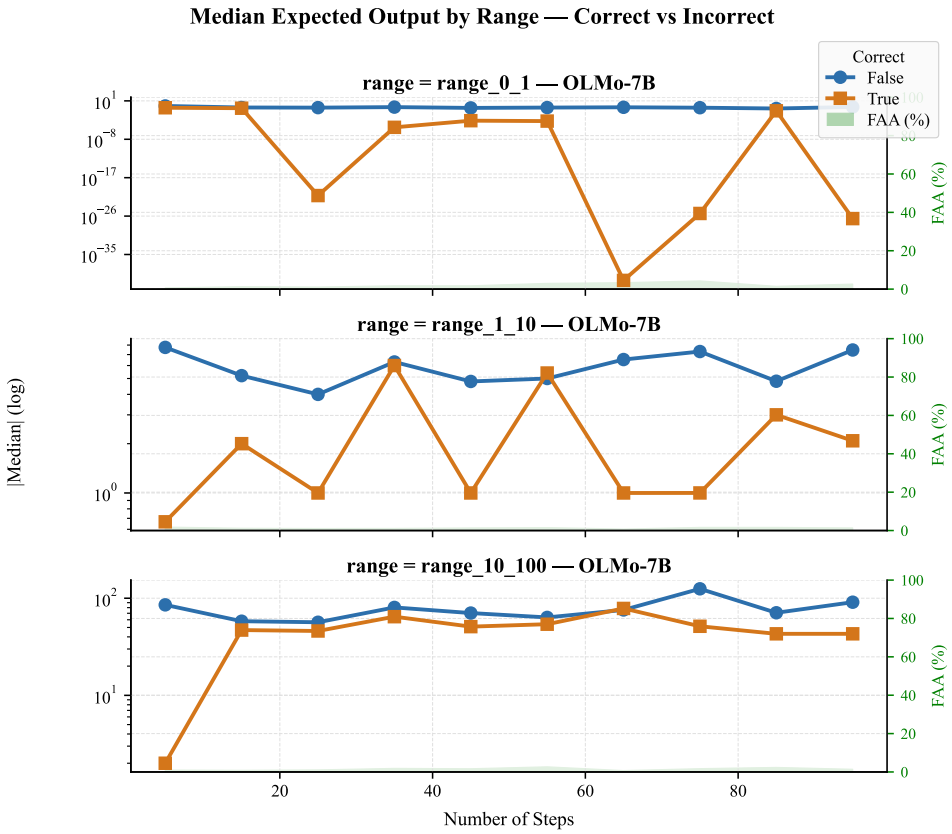


Figure 84: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). Across all ranges, performance tends to degrade for higher-magnitude expected outputs. This separation is more pronounced for the ranges [0,1] and [1,10] than for [10,100], suggesting that output magnitude may contribute to the observed performance decline.

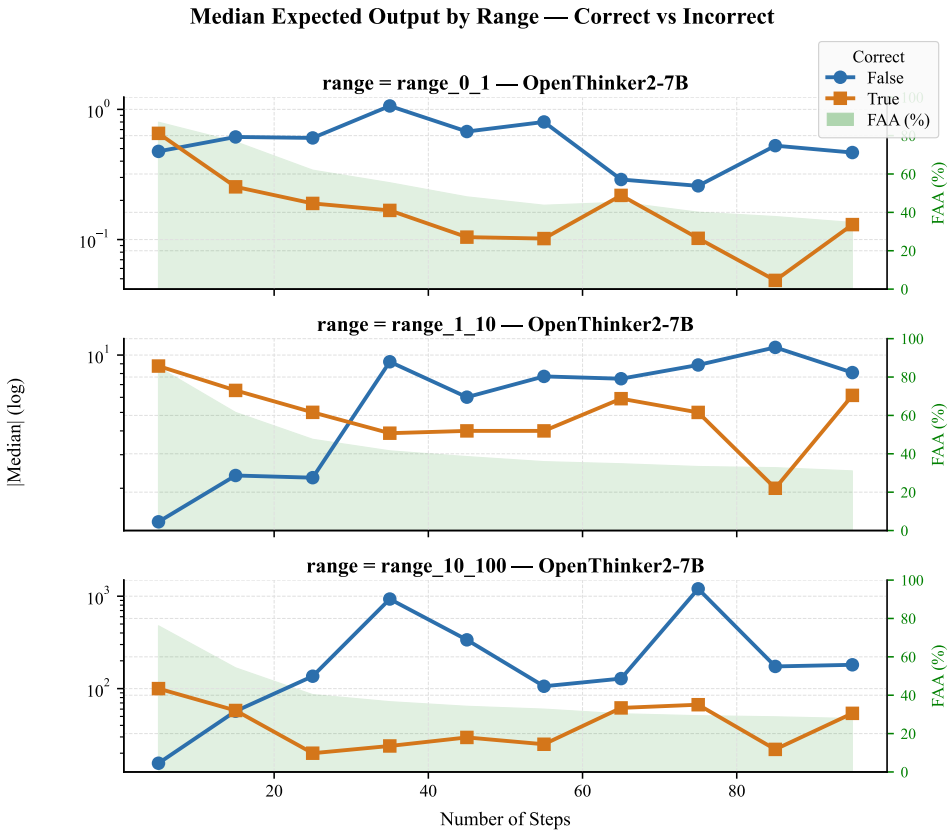


Figure 85: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). For all ranges of input, a noticeable separation in output magnitudes between correct and incorrect predictions is observed, where correct predictions tend to correspond to lower output magnitudes.

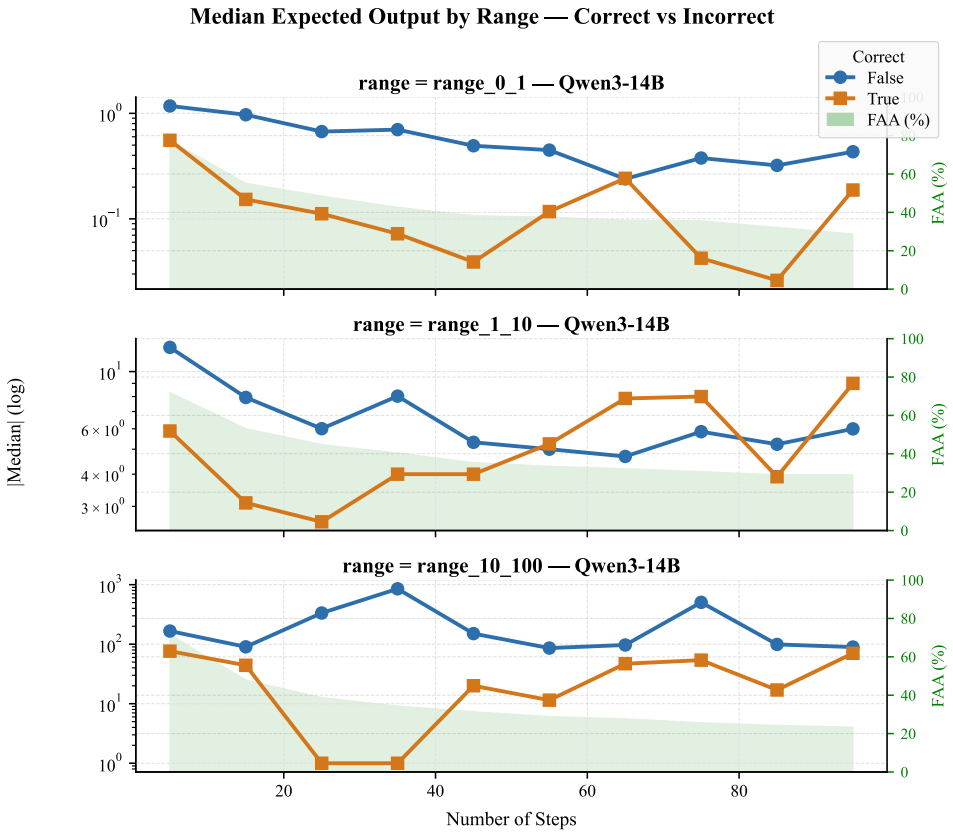


Figure 86: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). For all ranges of input, a noticeable separation in output magnitudes between correct and incorrect predictions is observed, where correct predictions tend to correspond to lower output magnitudes.

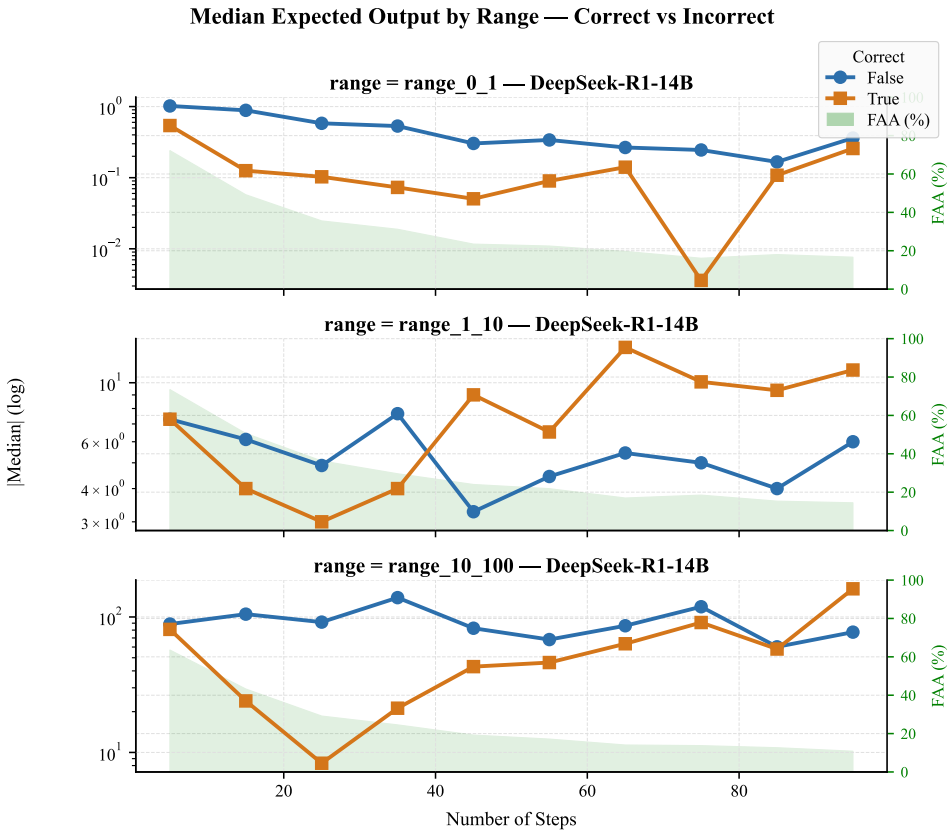


Figure 87: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). Across all ranges, a noticeable separation between the output magnitudes of correct and incorrect predictions is observed. For the ranges [0,1] and [10,100], correct predictions tend to be associated with lower output magnitudes. In contrast, for the range [1,10], the trend is reversed, with correct predictions increasingly associated with higher output magnitudes as the number of steps increases.

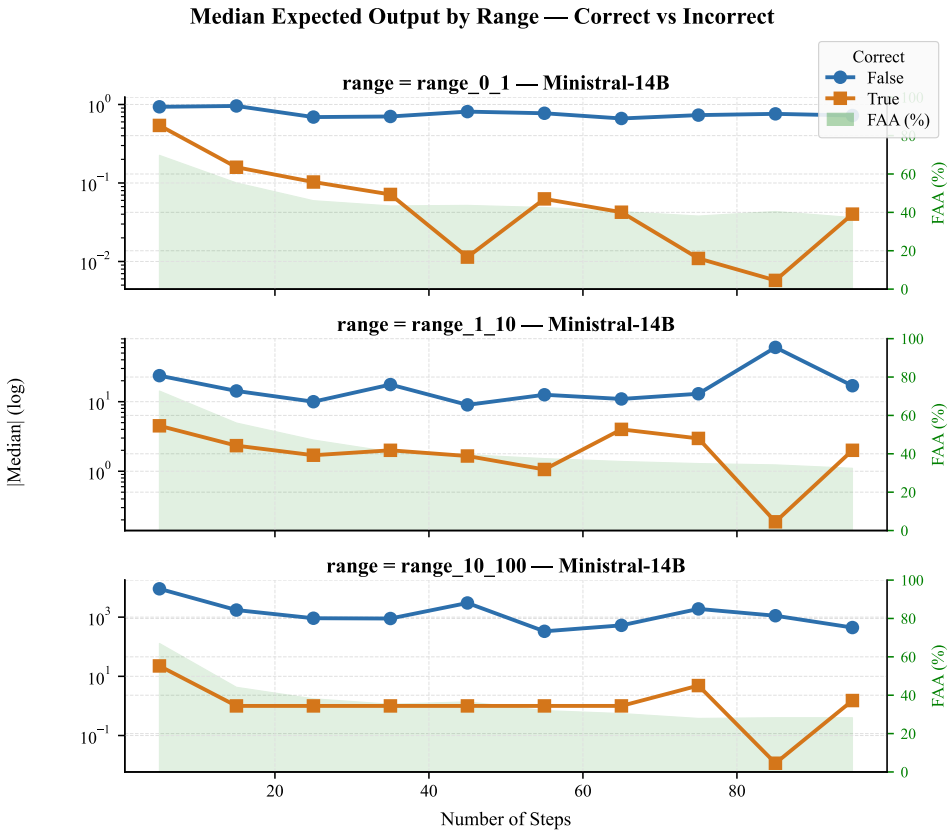


Figure 88: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). For all ranges of input, a noticeable separation in output magnitudes between correct and incorrect predictions is observed, where correct predictions tend to correspond to lower output magnitudes.

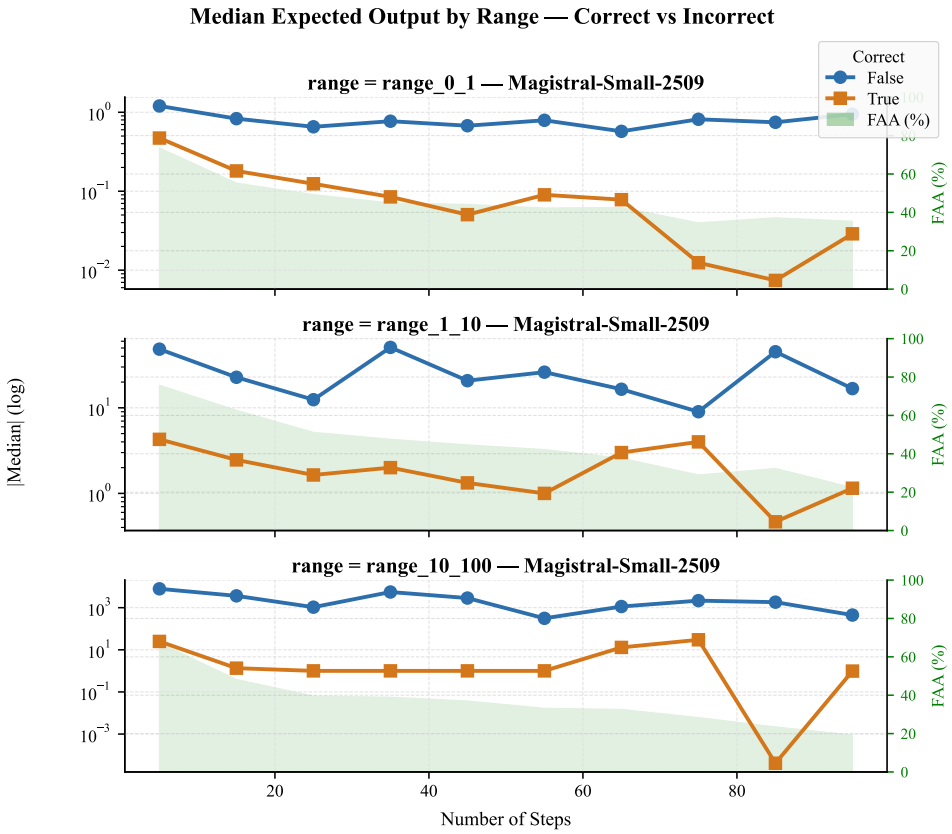


Figure 89: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). For all ranges of input, a noticeable separation in output magnitudes between correct and incorrect predictions is observed, where correct predictions tend to correspond to lower output magnitudes.

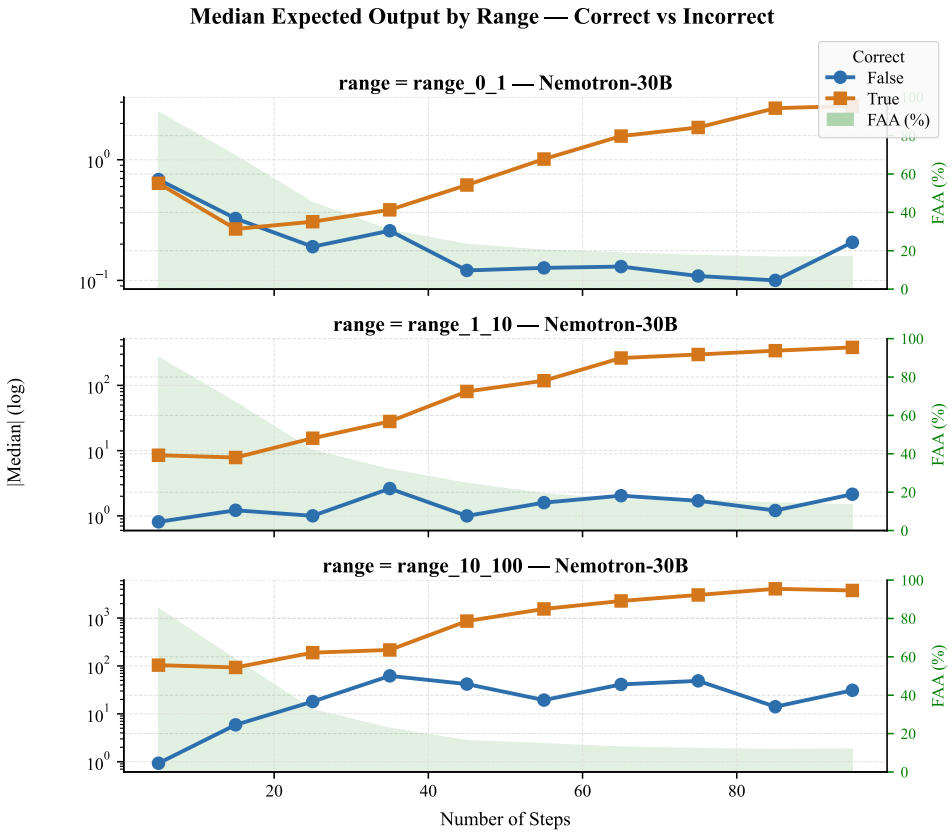


Figure 90: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). For all ranges of input, a noticeable separation in output magnitudes between correct and incorrect predictions is observed, where correct predictions tend to correspond to lower output magnitudes.

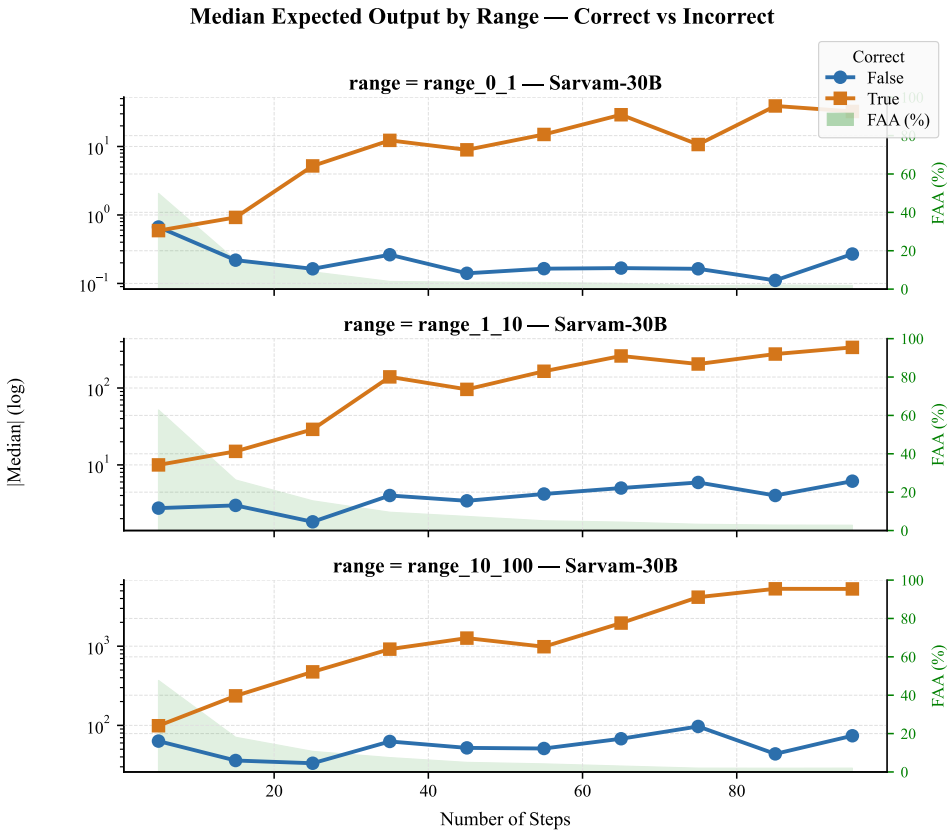


Figure 91: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). Across all input ranges, a noticeable separation in output magnitudes between correct and incorrect predictions is observed, with correct predictions generally associated with higher output magnitudes. Notably, this pattern contrasts with observations from other models, where correct predictions are often associated with lower output magnitudes.

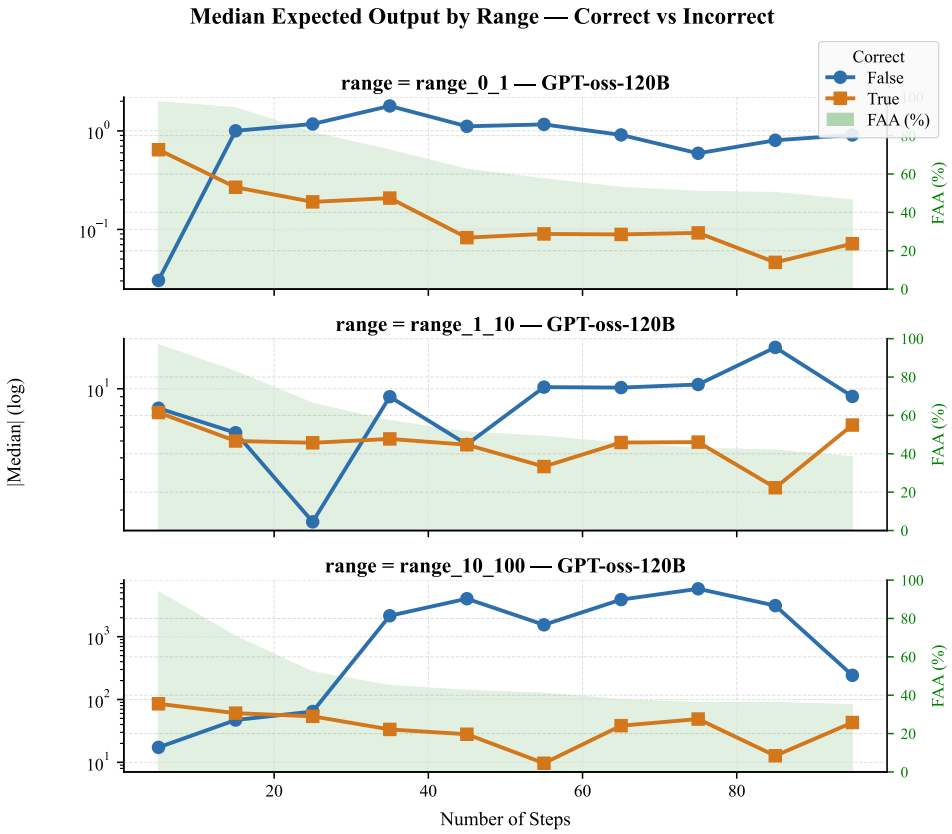


Figure 92: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). For all ranges of input, a noticeable separation in output magnitudes between correct and incorrect predictions is observed, where correct predictions tend to correspond to lower output magnitudes.

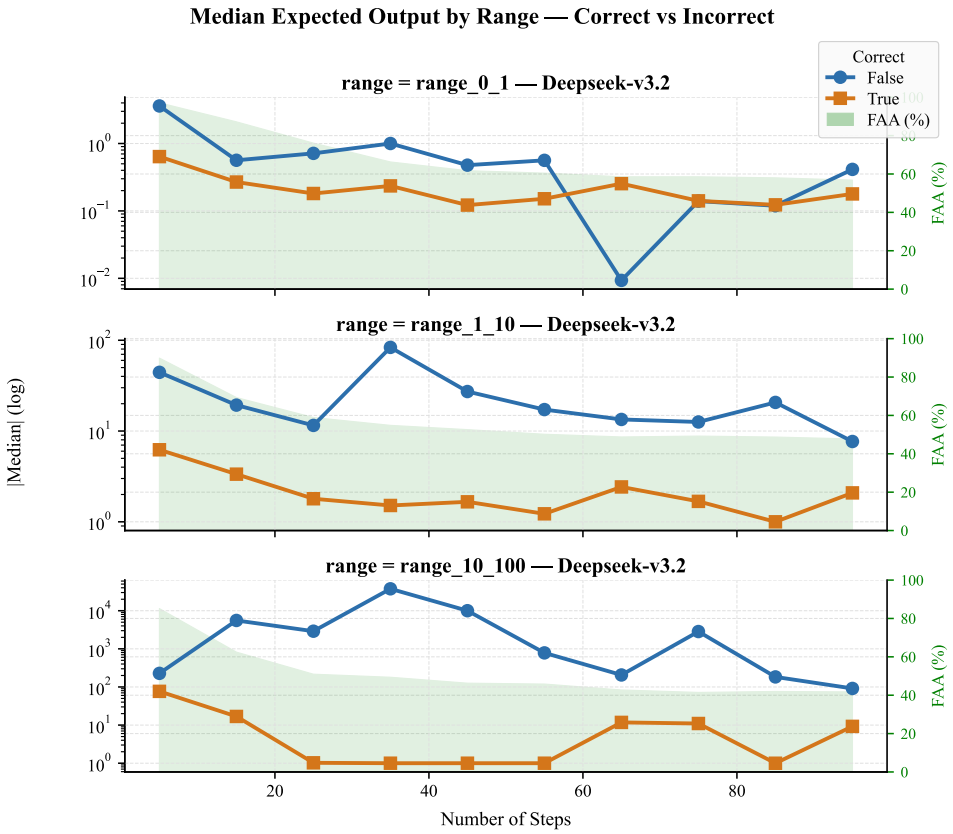


Figure 93: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). For all ranges of input, a noticeable separation in output magnitudes between correct and incorrect predictions is observed, where correct predictions tend to correspond to lower output magnitudes.

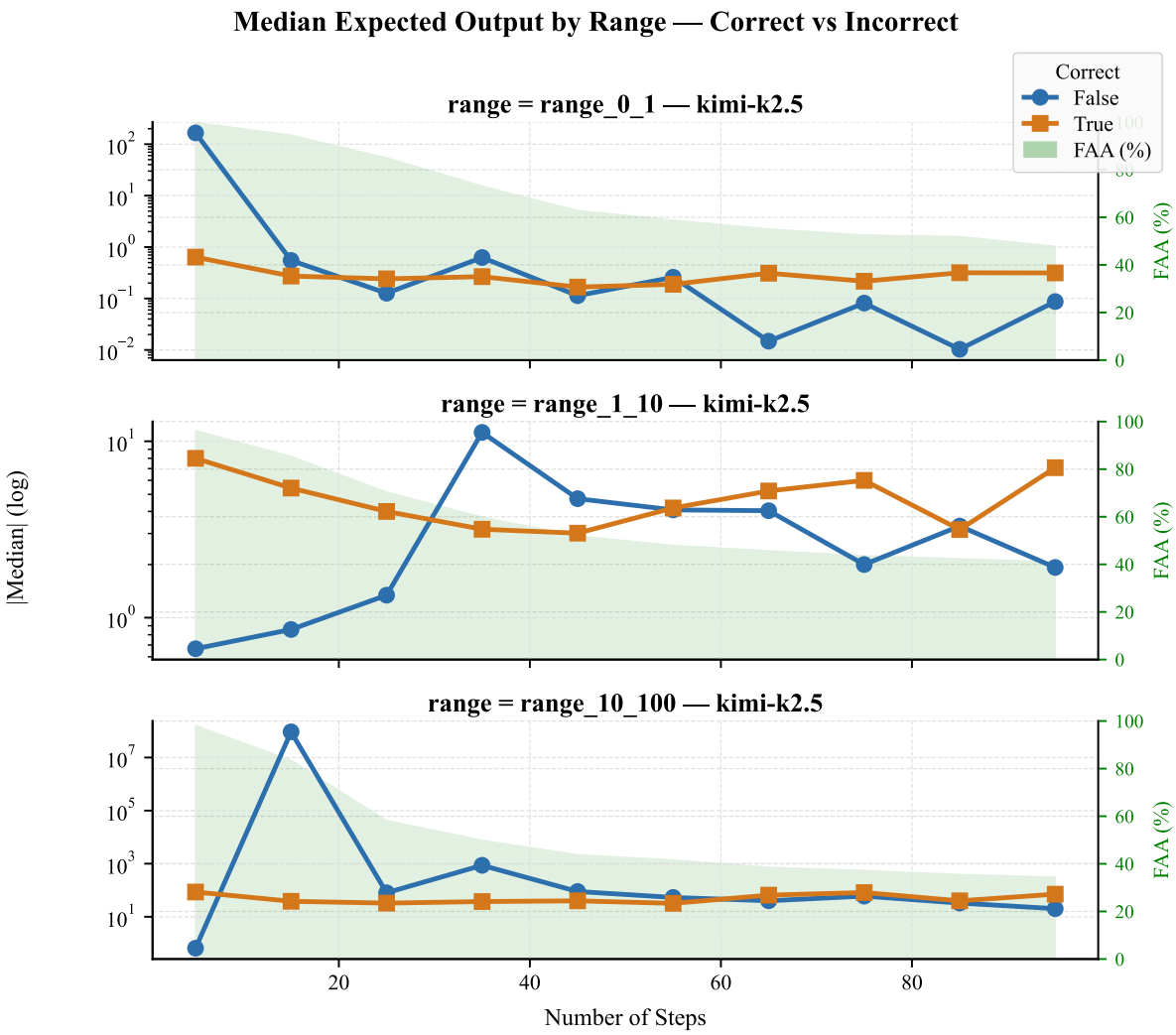


Figure 94: Median expected output across procedure lengths for different input ranges ([0,1], [1,10], [10,100]), separated by correct and incorrect predictions (log scale). We cannot se a clear separation between output magnitude of correct and incorrect prediction.

G.6 Median expected output across Task type

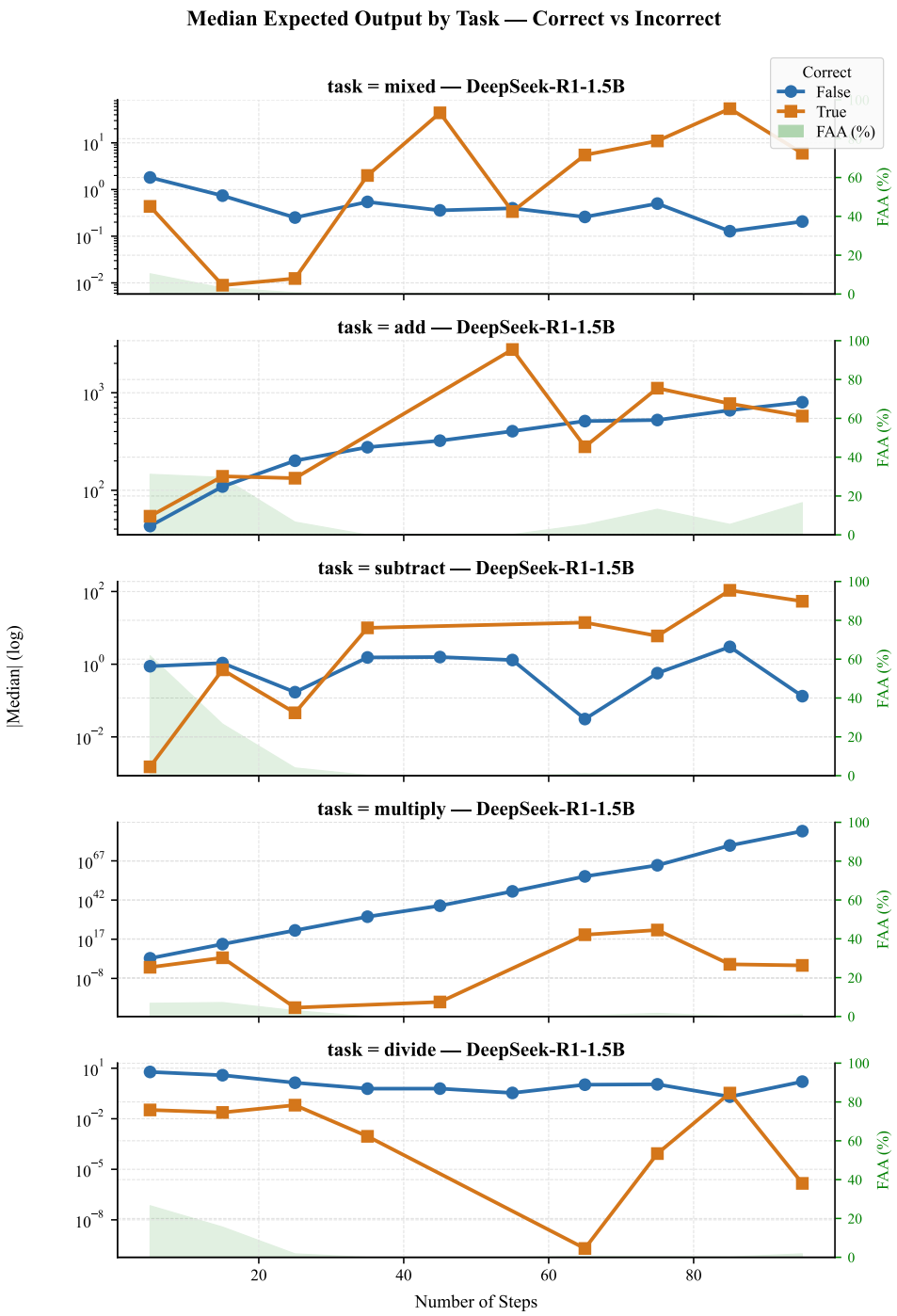


Figure 95: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). Output magnitudes vary substantially across tasks, with multiplication and division exhibiting significantly larger scales than addition and subtraction. For multiplication (and partially for division), the gap between correct and incorrect predictions increases at higher step counts, suggesting a divergence in output magnitude. However, this pattern is not consistently observed across all tasks or steps.

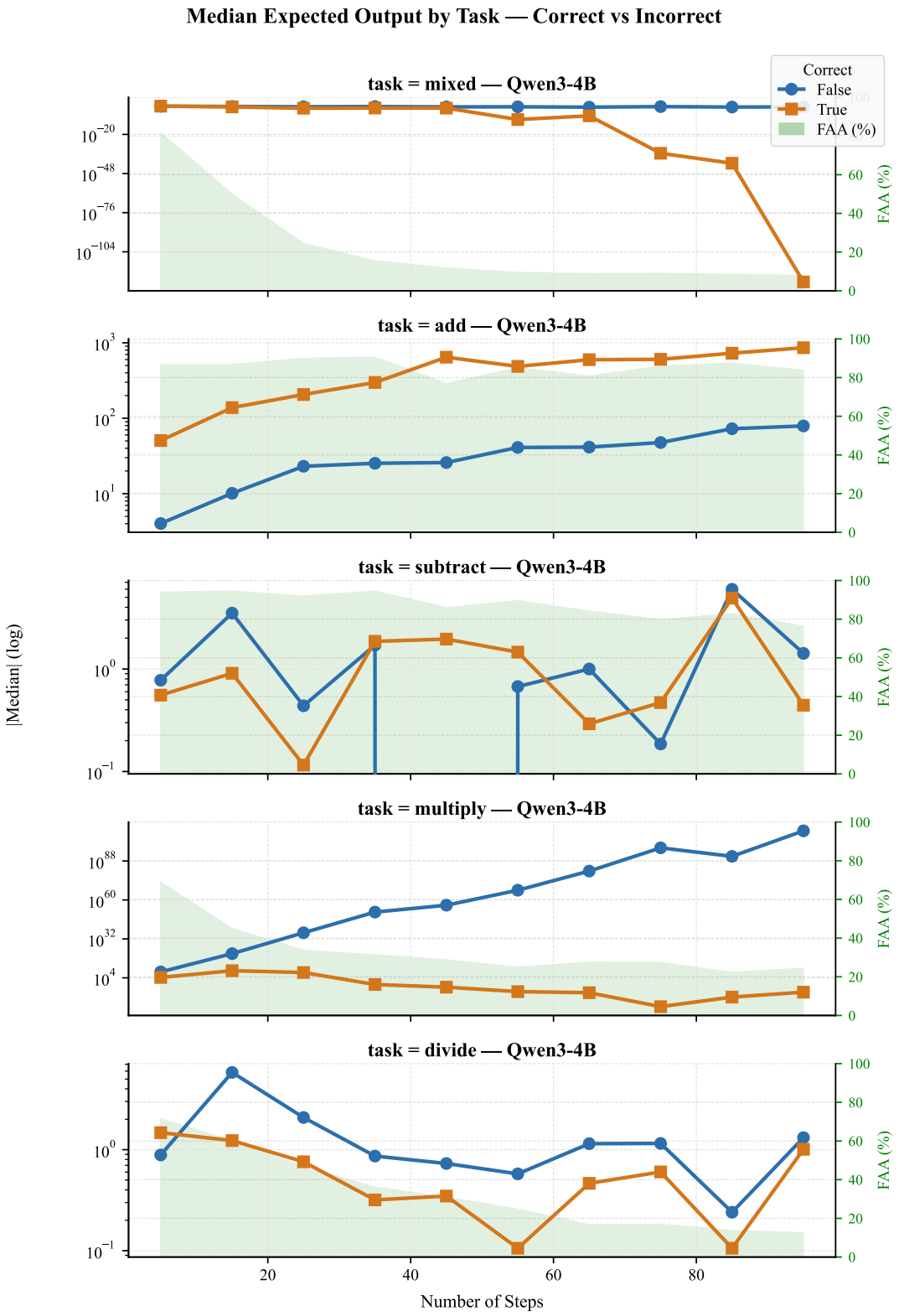


Figure 96: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). The relationship between output magnitude and correctness varies across tasks. For multiplication, the separation between correct and incorrect predictions increases with step length, with correct predictions generally associated with lower output magnitudes; a similar but less consistent trend is observed for division. In contrast, for addition, correct predictions tend to correspond to higher output magnitudes. For subtraction, the distributions largely overlap, indicating no clear separation. For mixed tasks, limited separation is observed at shorter step lengths, while a more noticeable divergence emerges at larger step counts.

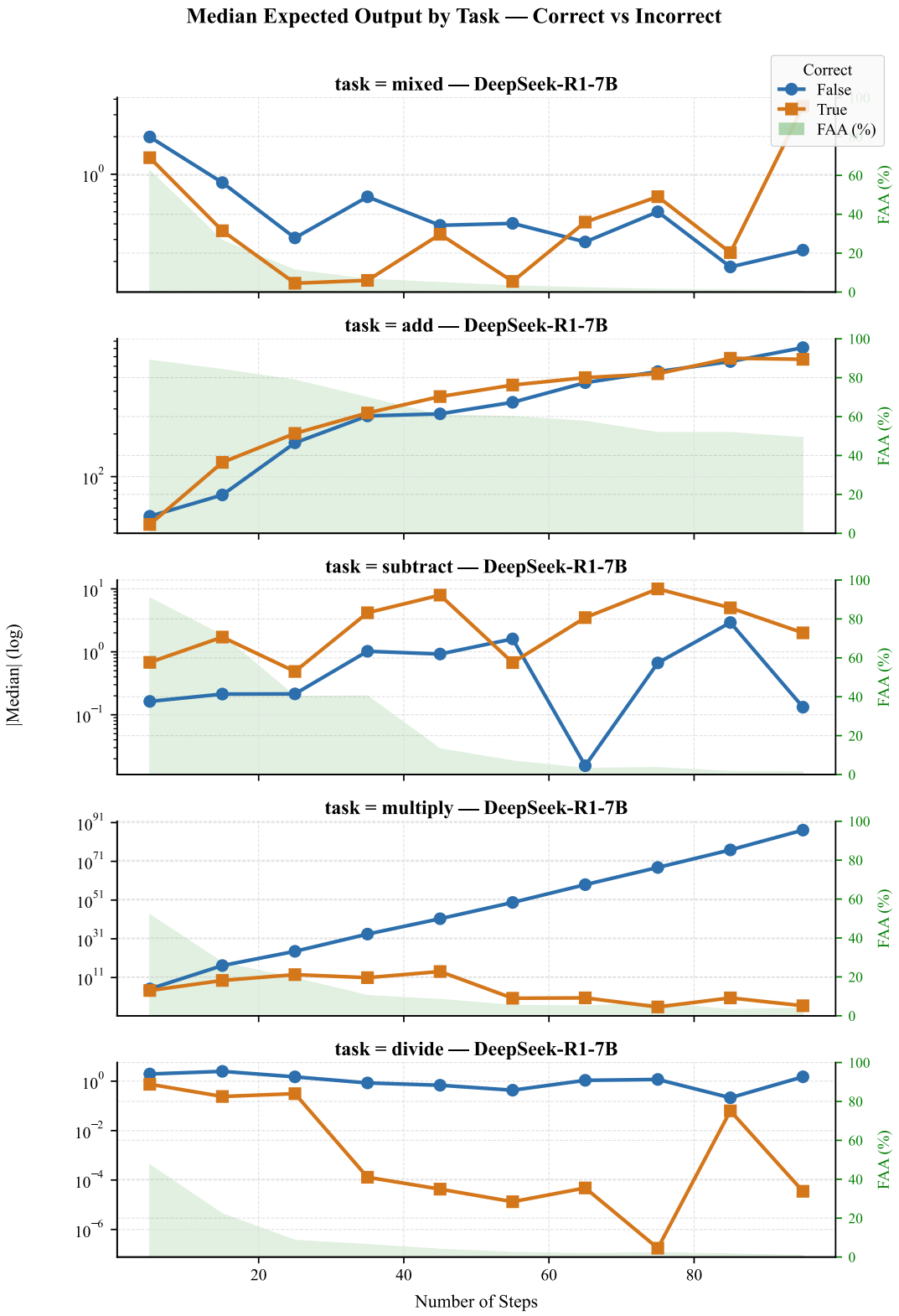


Figure 97: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). The relationship between output magnitude and correctness varies across tasks. For multiplication, the separation between correct and incorrect predictions increases with step length, with correct predictions generally associated with lower output magnitudes; a similar but less consistent pattern is observed for division. For addition, subtraction, and mixed tasks, the distributions largely overlap, indicating no clear separation. These observations suggest that output magnitude may contribute to the lower performance observed in multiplication and division, though it does not fully explain the behavior across all tasks.

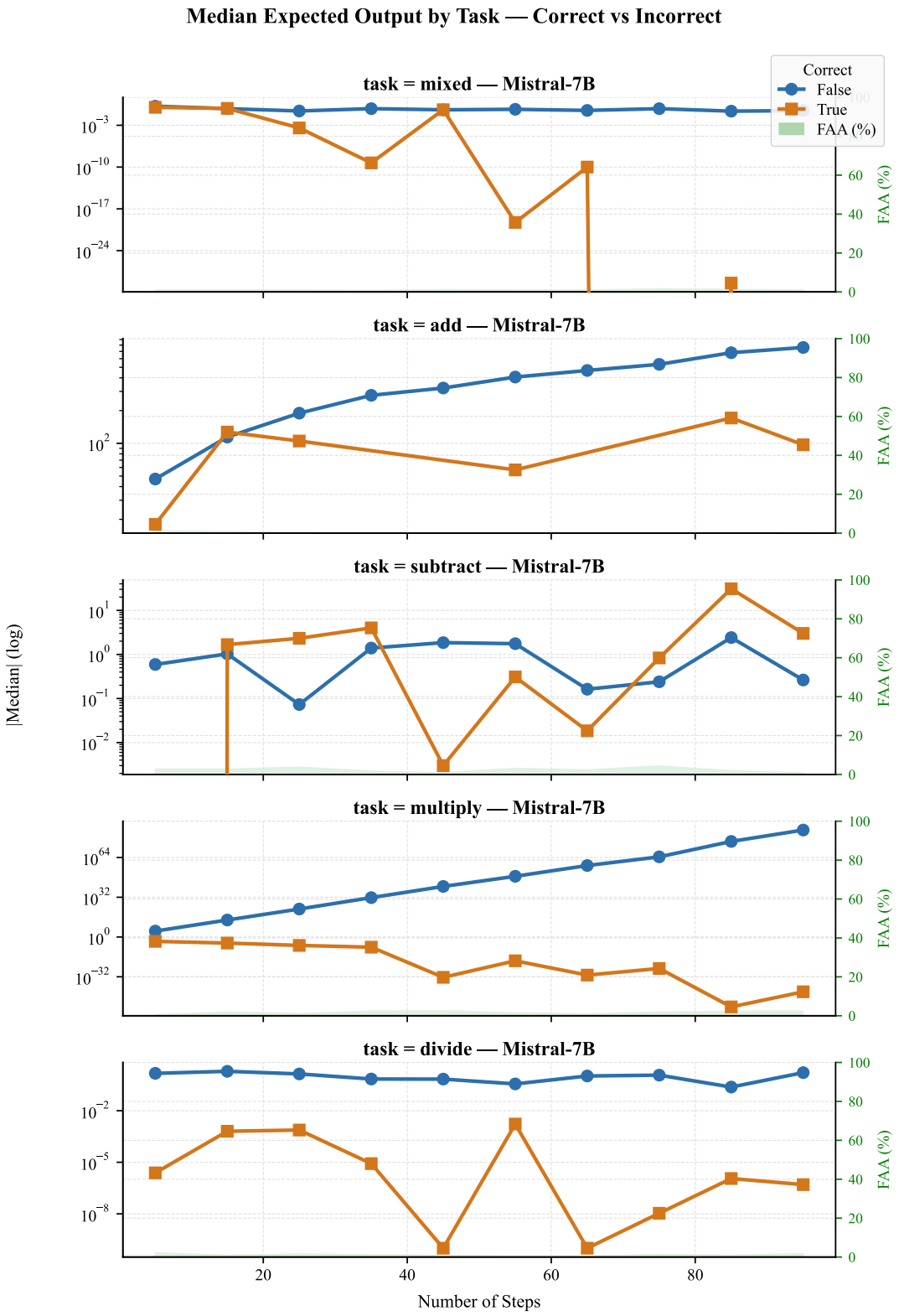


Figure 98: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). The relationship between output magnitude and correctness varies across tasks. For multiplication, the separation between correct and incorrect predictions increases with step length, with correct predictions generally associated with lower output magnitudes; a similar but less consistent trend is observed for addition and division. For subtraction, the distributions largely overlap, indicating no clear separation. For mixed tasks, limited separation is observed at shorter step lengths, while a more noticeable divergence emerges at larger step counts.

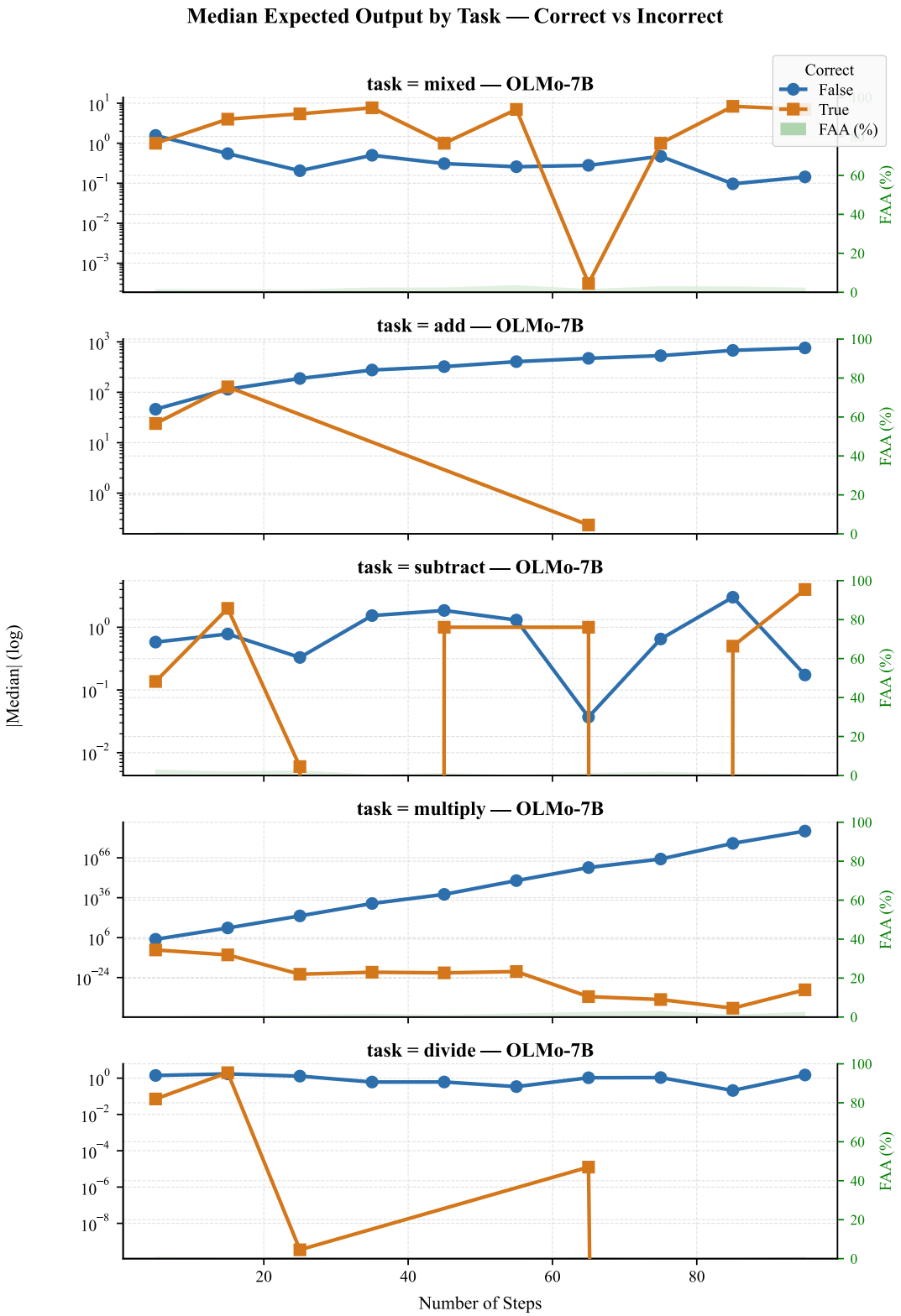


Figure 99: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). The relationship between output magnitude and correctness varies across tasks. For addition and multiplication, the separation between correct and incorrect predictions increases with step length, with correct predictions generally associated with lower output magnitudes; a similar but less consistent trend is observed for division. For subtraction and mixed task, the distributions largely overlap, indicating no clear separation.

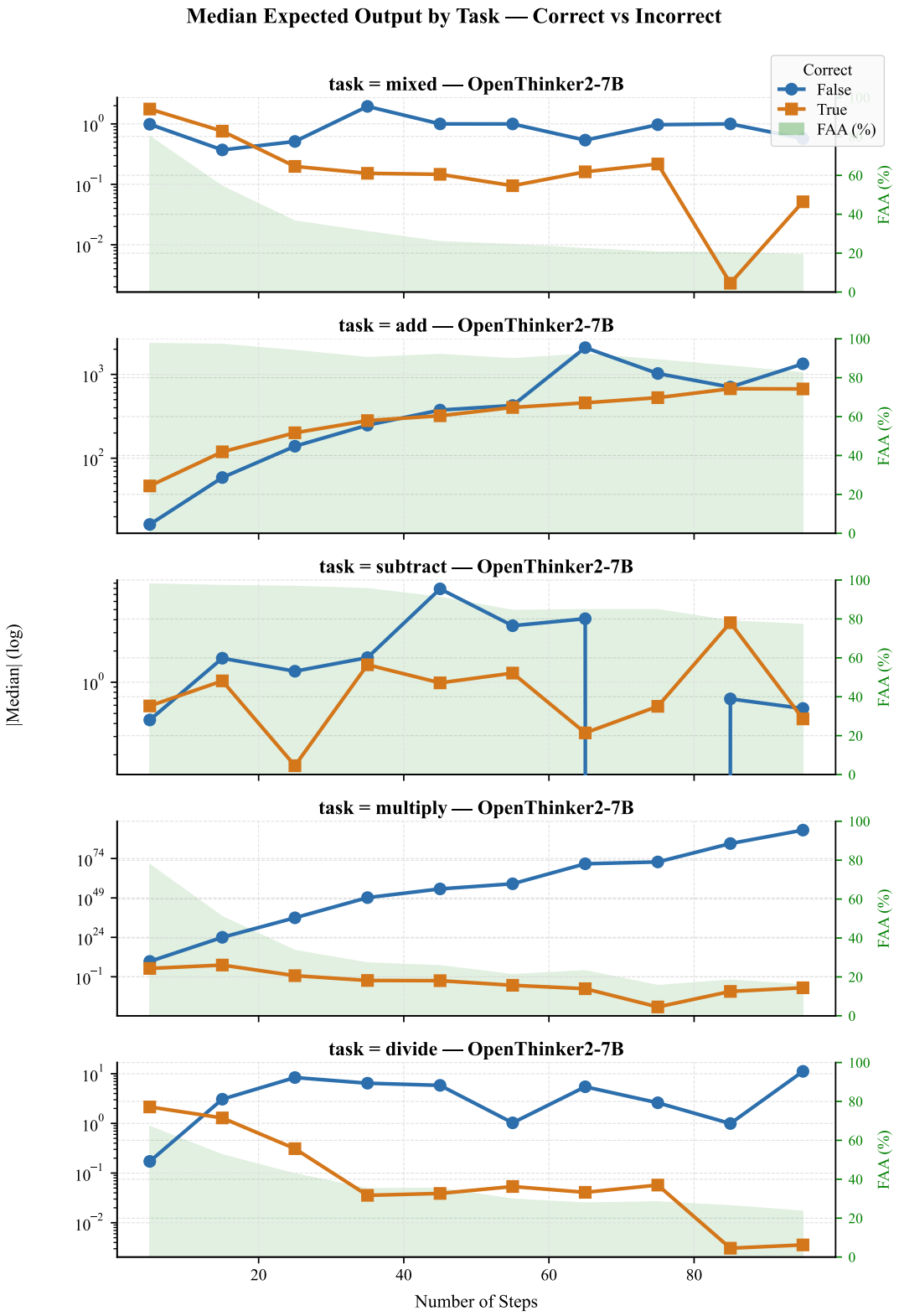


Figure 100: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). The relationship between output magnitude and correctness varies across tasks. For multiplication and division, the separation between correct and incorrect predictions increases with step length, with correct predictions generally associated with lower output magnitudes; a similar but less consistent trend is observed for mixed task. For addition and subtraction, the distributions largely overlap, indicating no clear separation.

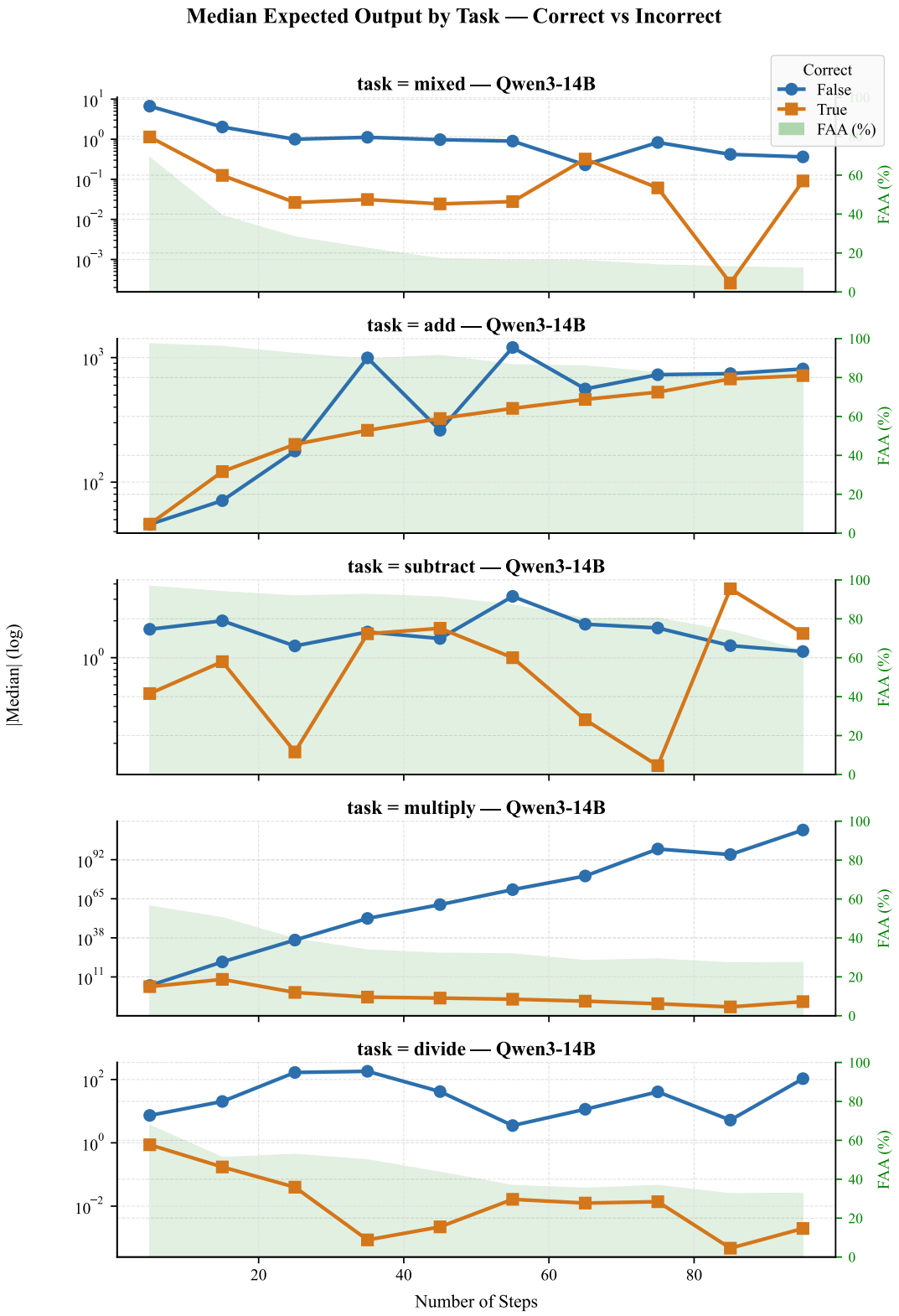


Figure 101: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). The relationship between output magnitude and correctness varies across tasks. For multiplication and division, the separation between correct and incorrect predictions increases with step length, with correct predictions generally associated with lower output magnitudes; a similar but less consistent trend is observed for mixed tasks. For addition, correct predictions tend to be associated with higher output magnitudes, indicating a distinct but opposite trend. For subtraction, the distributions show substantial overlap with no consistent separation.

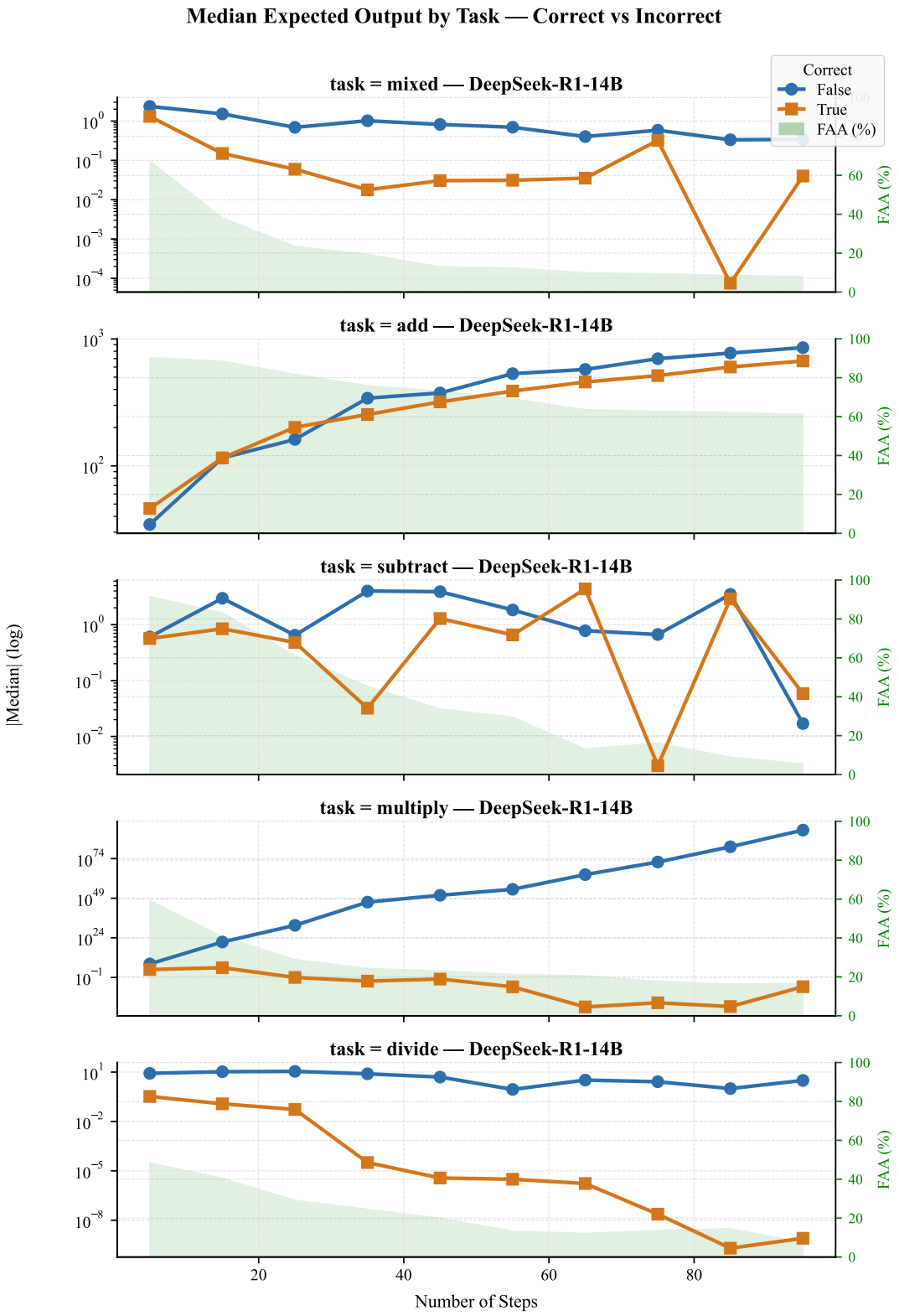


Figure 102: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). The relationship between output magnitude and correctness varies across tasks. For multiplication and division, the separation between correct and incorrect predictions increases with step length, with correct predictions associated with lower output magnitudes; a similar but less consistent trend is observed for mixed task. For addition and subtraction, the distributions largely overlap, indicating no clear separation.

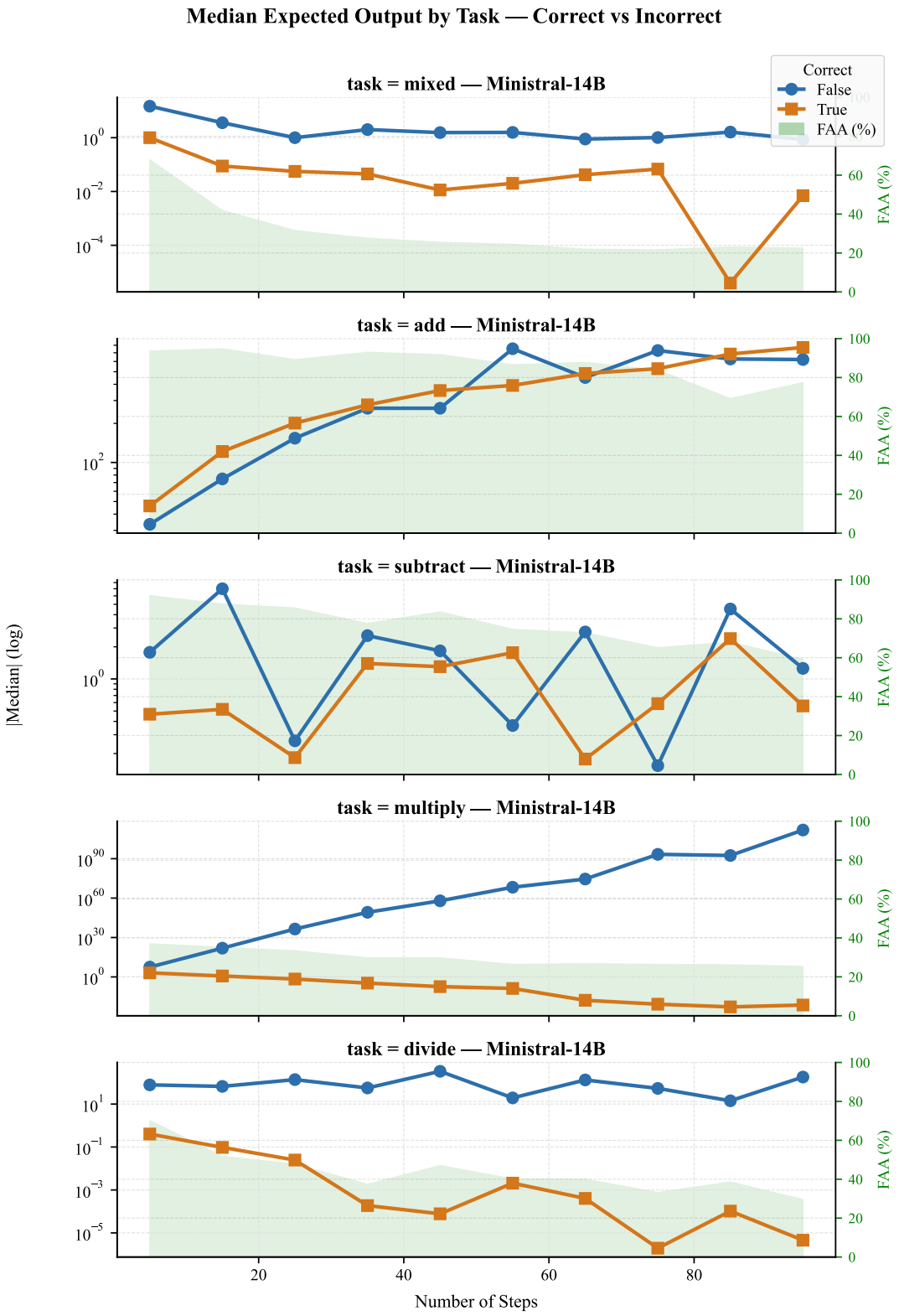


Figure 103: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). The relationship between output magnitude and correctness varies across tasks. For multiplication and division, the separation between correct and incorrect predictions increases with step length, with correct predictions associated with lower output magnitudes; a similar but less consistent trend is observed for mixed task. For addition and subtraction, the distributions largely overlap, indicating no clear separation.

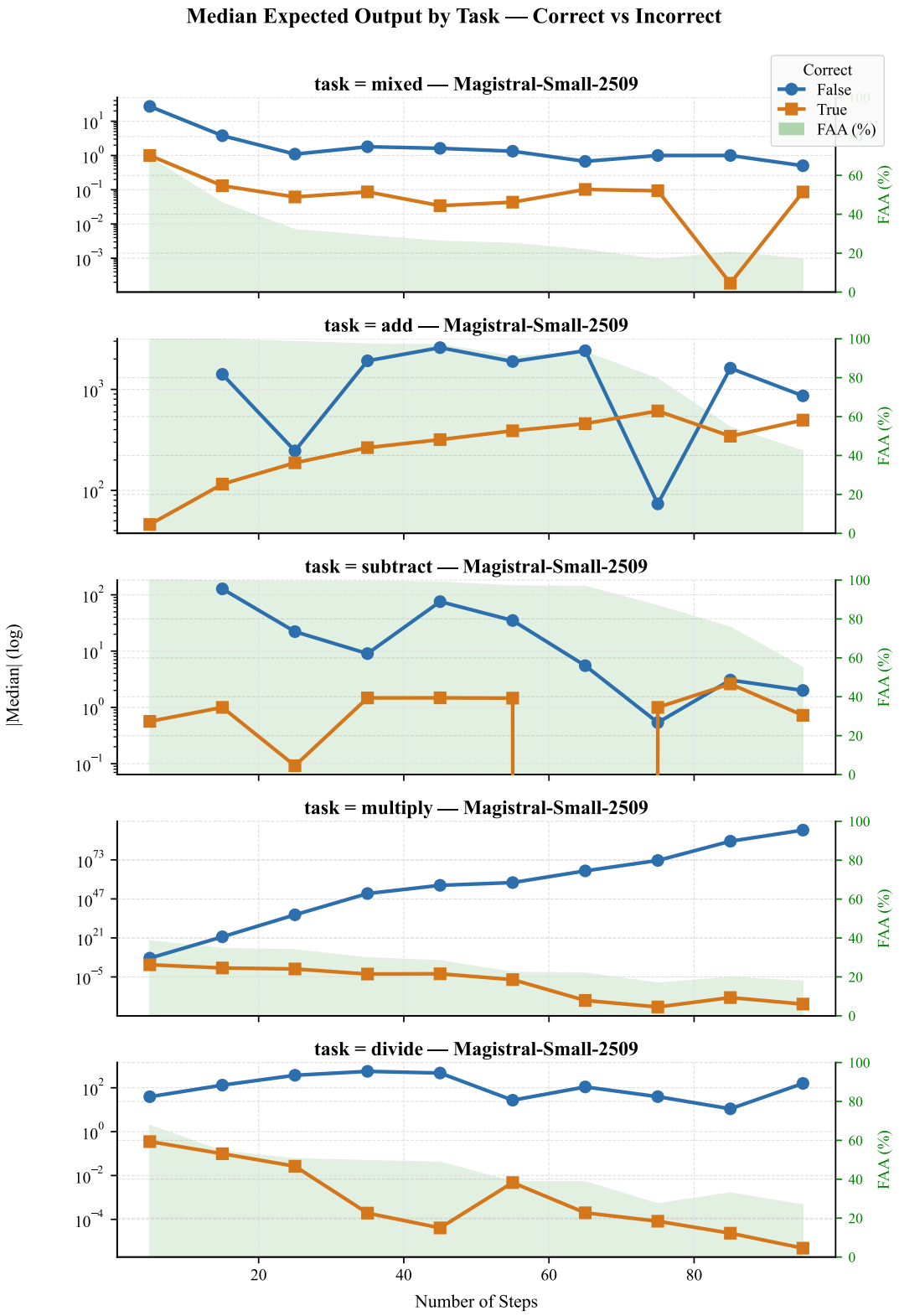


Figure 104: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). The relationship between output magnitude and correctness varies across tasks. For multiplication and division, the separation between correct and incorrect predictions increases with step length, with correct predictions associated with lower output magnitudes; a similar but less consistent trend is observed for addition, subtraction and mixed task.

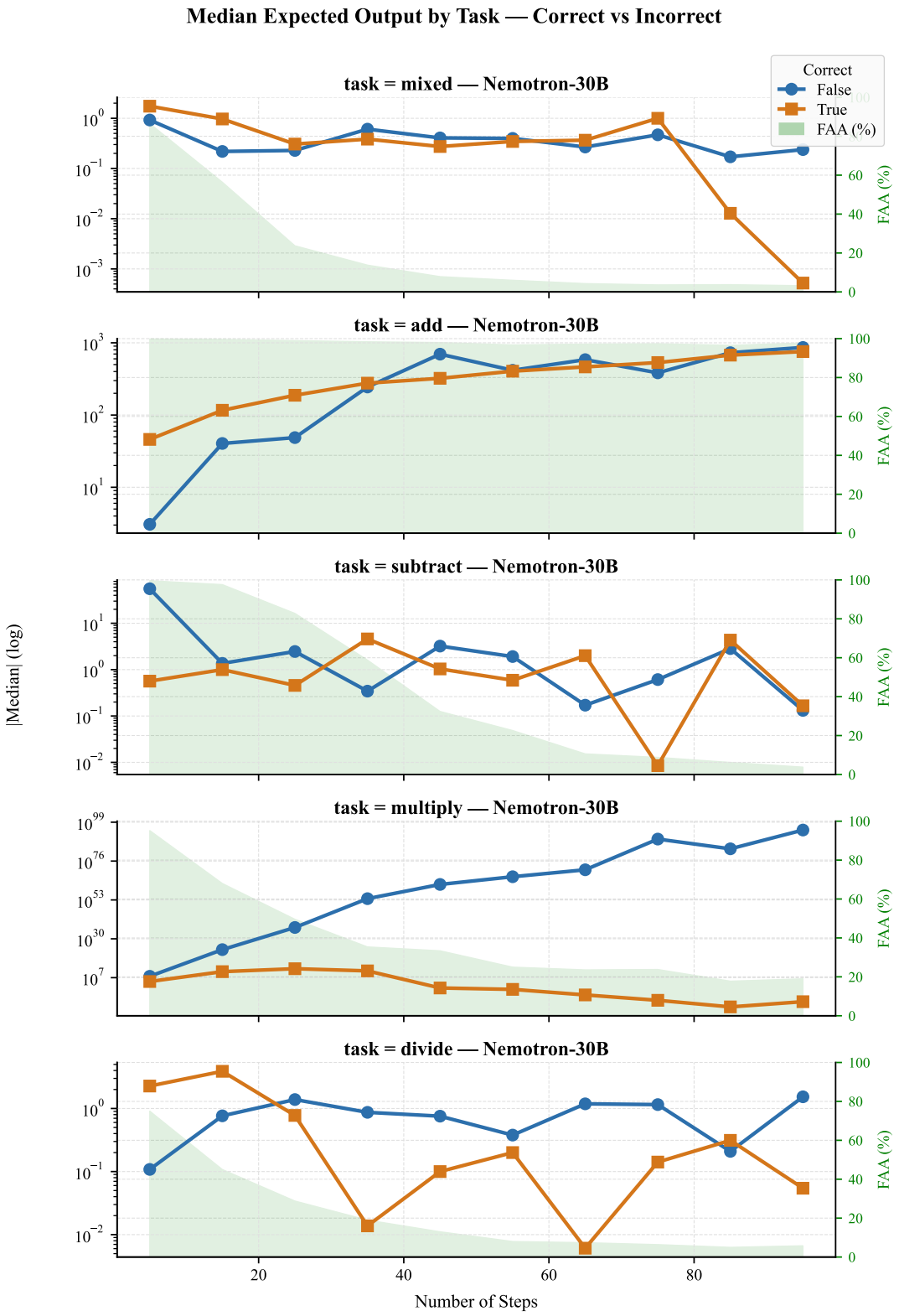


Figure 105: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). The relationship between output magnitude and correctness varies across tasks. For multiplication, the separation between correct and incorrect predictions increases with step length, with correct predictions generally associated with lower output magnitudes; a similar but less consistent trend is observed for division. For addition, subtraction and mixed task, the distributions largely overlap, indicating no clear separation.

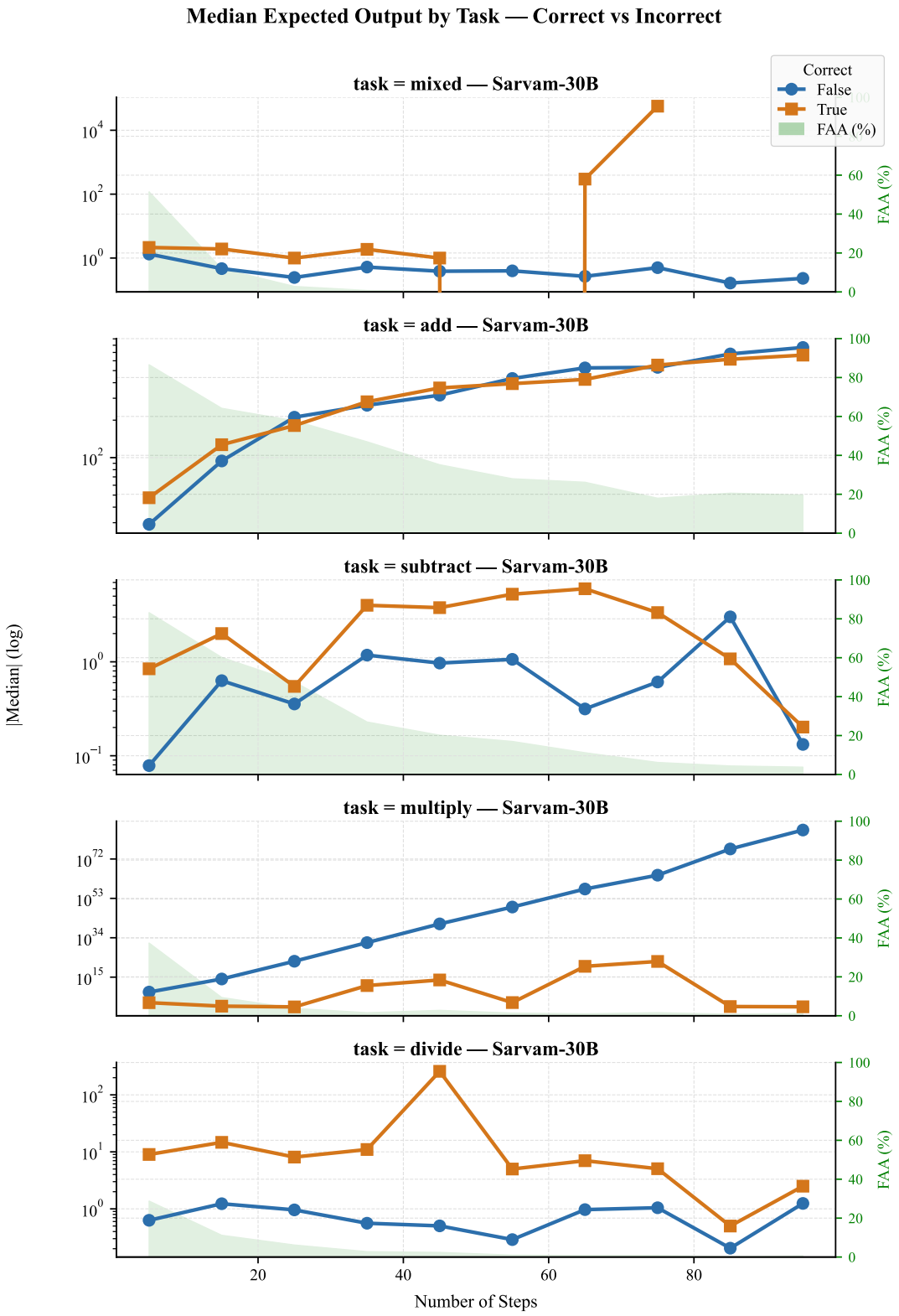


Figure 106: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). The relationship between output magnitude and correctness varies across tasks. For multiplication, the separation between correct and incorrect predictions increases with step length, with correct predictions associated with lower output magnitudes. In contrast, for division, an opposite trend is observed, where correct predictions are associated with higher output magnitudes; a similar but less consistent pattern is observed for subtraction. For addition and mixed tasks, the distributions largely overlap, indicating no clear separation.

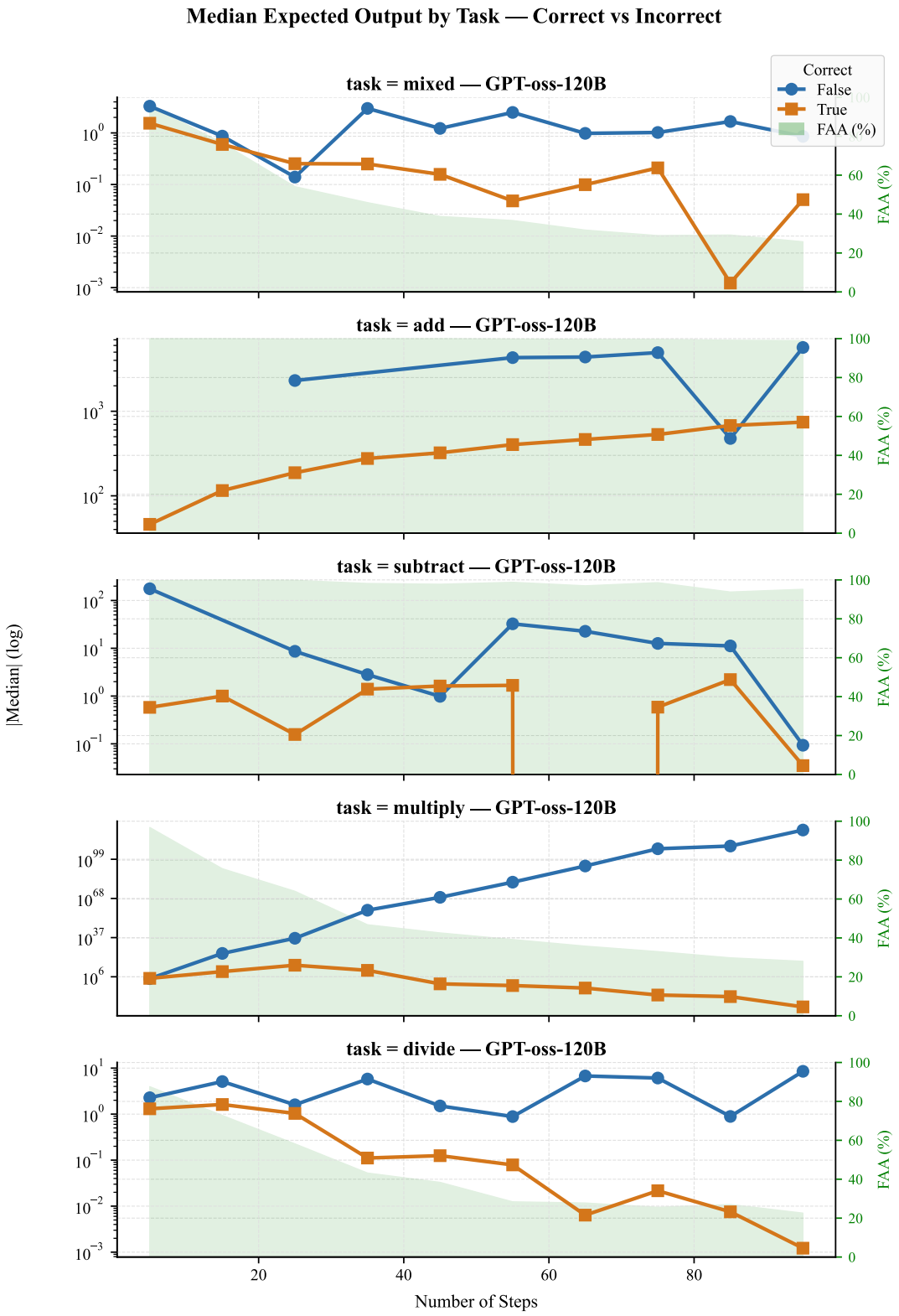


Figure 107: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). The relationship between output magnitude and correctness varies across tasks. For multiplication and division, the separation between correct and incorrect predictions increases with step length, with correct predictions generally associated with lower output magnitudes; a similar but less consistent trend is observed for addition. subtraction and mixed task.

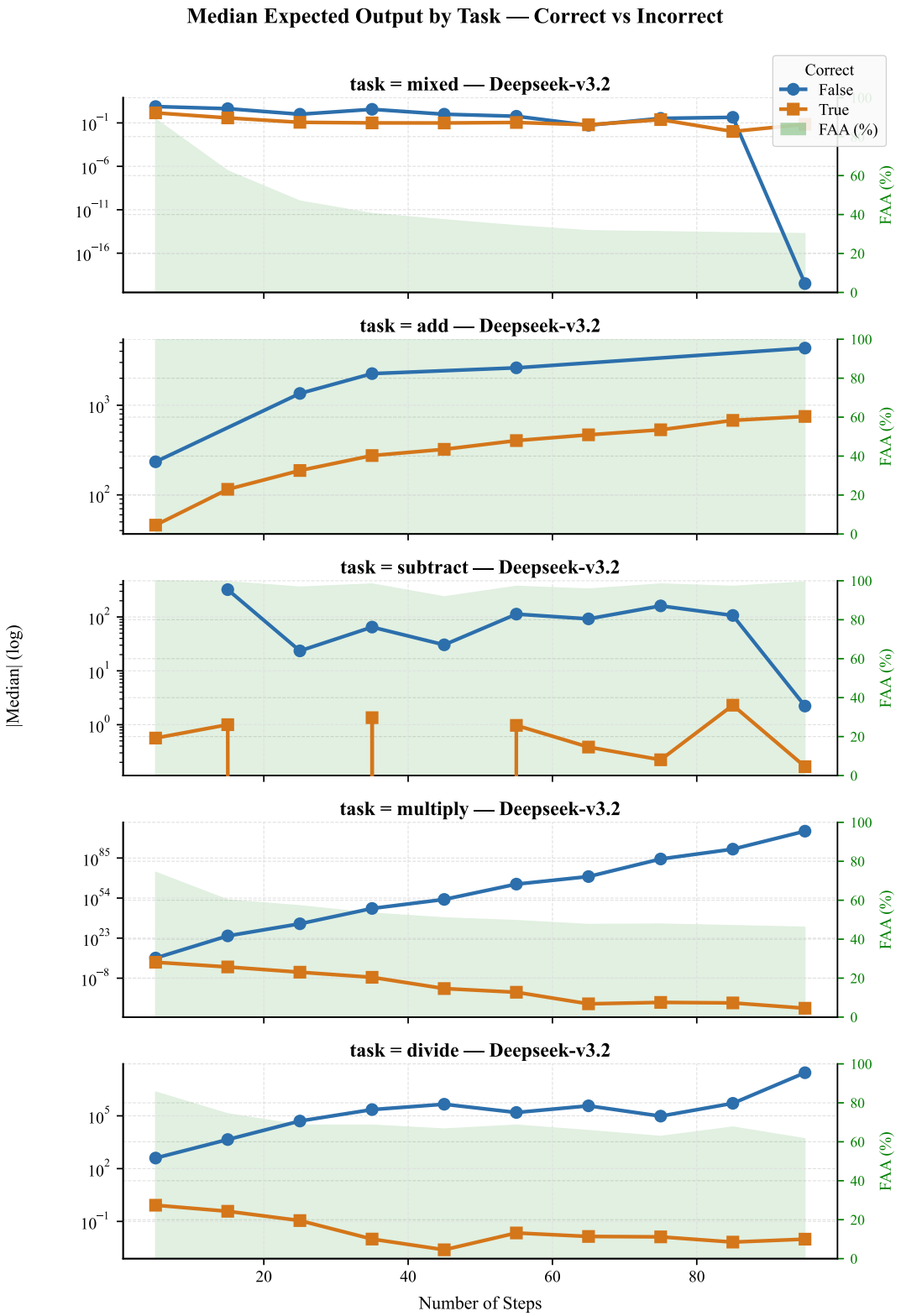


Figure 108: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). The relationship between output magnitude and correctness varies across tasks. For addition, subtraction, multiplication and division, the separation between correct and incorrect predictions increases with step length, with correct predictions generally associated with lower output magnitudes; a similar but less consistent trend is observed for mixed task.

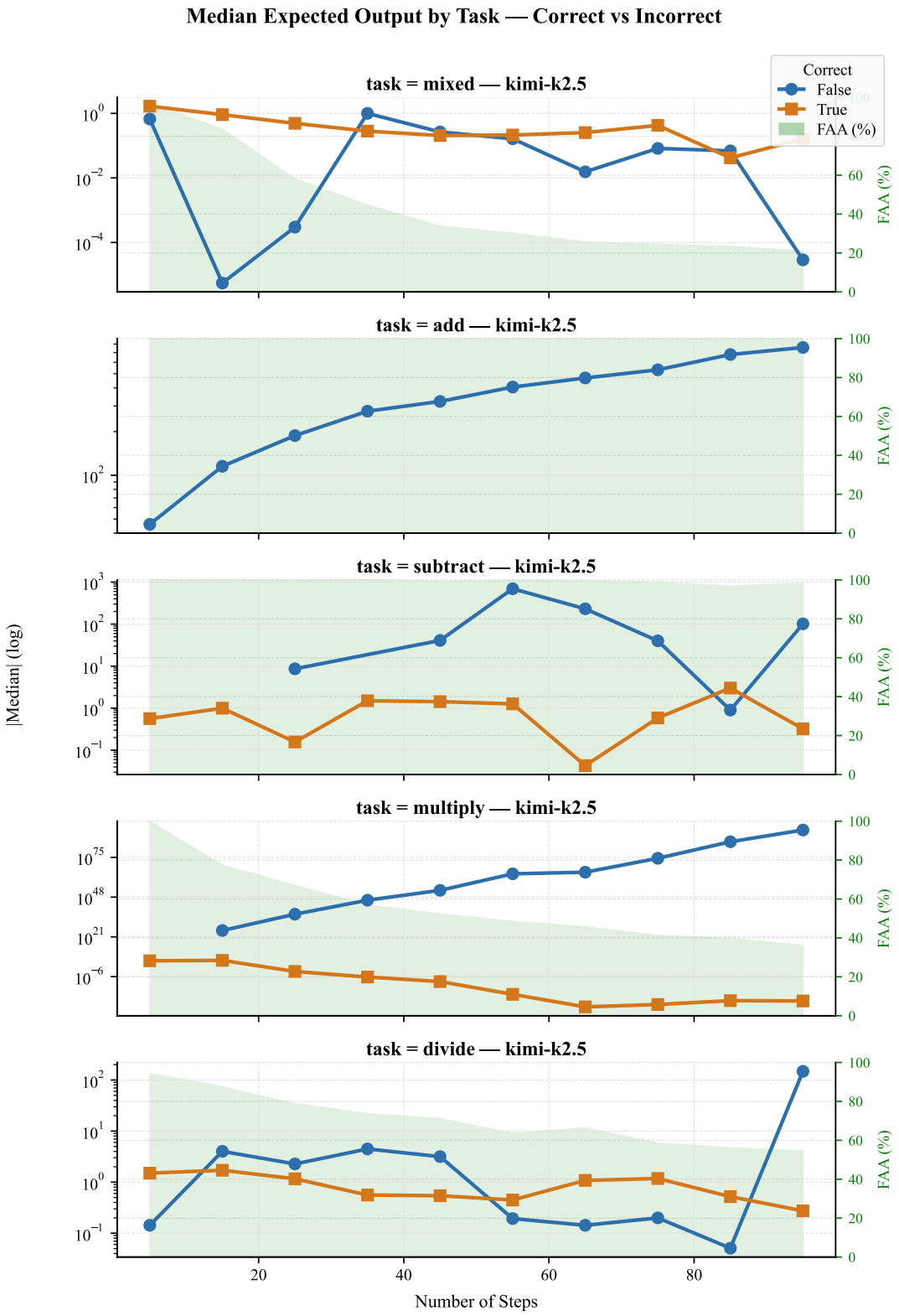


Figure 109: Median expected output across procedure lengths for different task types (addition, subtraction, multiplication, division, and mixed), grouped by correctness (log scale). We observed the separation between correct and incorrect predictions increases with step length for multiplication task, where as for Adddition, Subtraction, Division and Mixed operation we observed a overlap between the trend line.

G.7 Step Analysis across procedure lengths.

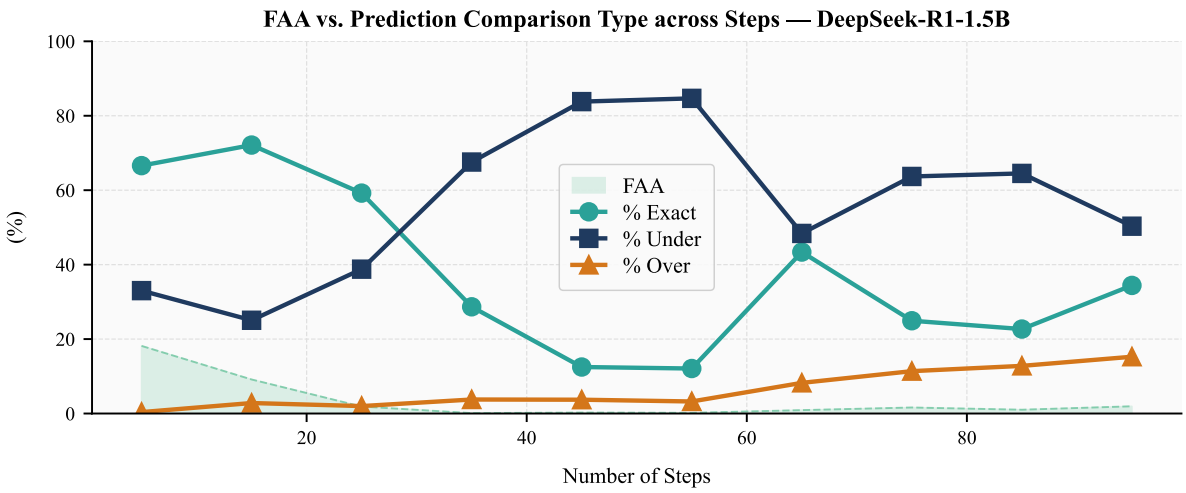


Figure 110: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). As the number of steps increases, overall FAA declines, accompanied by a decrease in exact-step executions and a substantial increase in under-execution. This suggests that longer tasks are associated with a tendency for the model to terminate execution prematurely, which correlates with reduced performance.

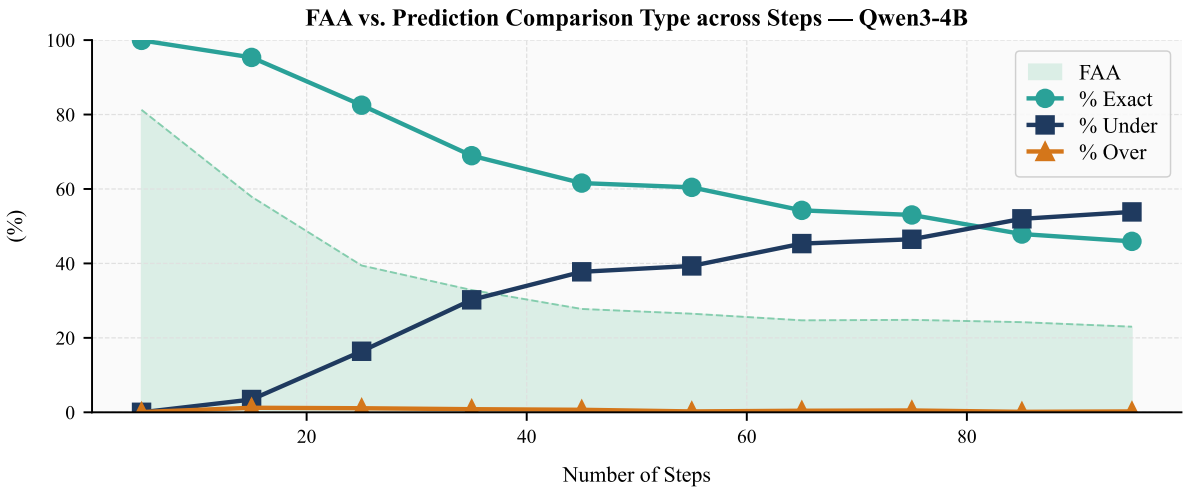


Figure 111: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). As the number of steps increases, overall FAA declines, accompanied by a decrease in exact-step executions and a substantial increase in under-execution. This suggests that longer tasks are associated with a tendency for the model to terminate execution prematurely, which correlates with reduced performance.

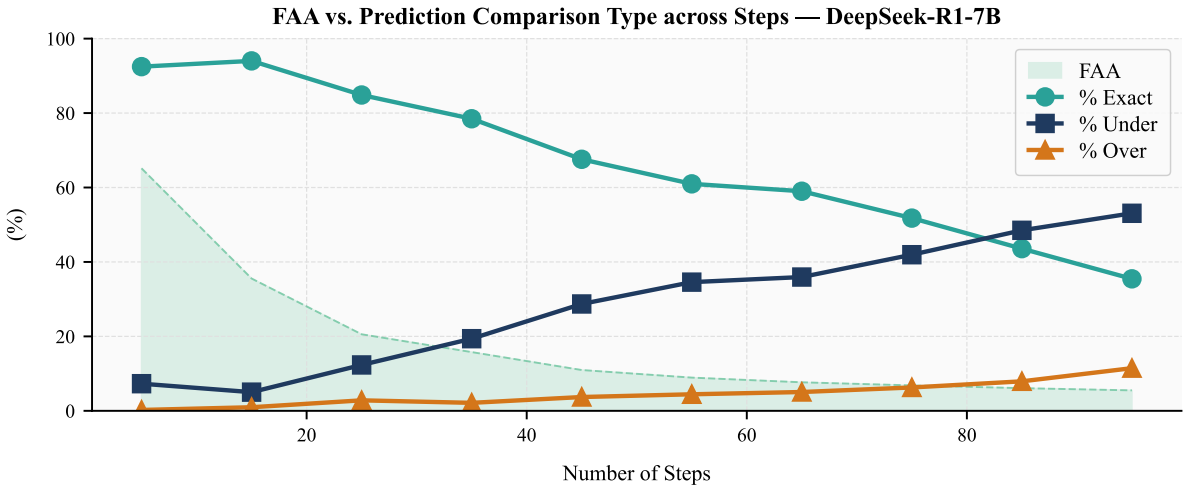


Figure 112: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). As the number of steps increases, overall FAA declines, accompanied by a decrease in exact-step executions and a substantial increase in under-execution. This suggests that longer tasks are associated with a tendency for the model to terminate execution prematurely, which correlates with reduced performance.

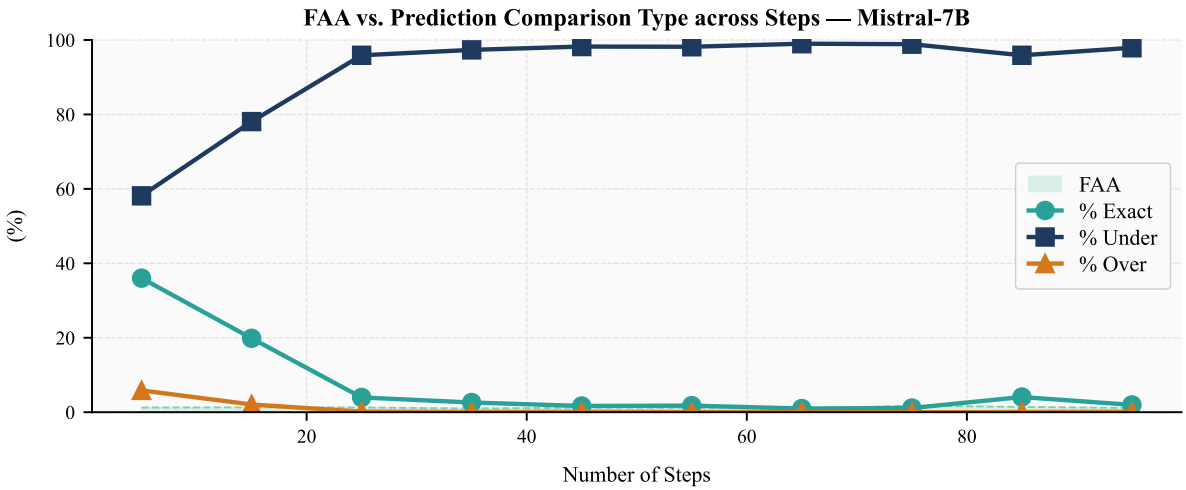


Figure 113: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). As algorithm length increases, the model increasingly under-executes steps, which is associated with the observed decline in performance.

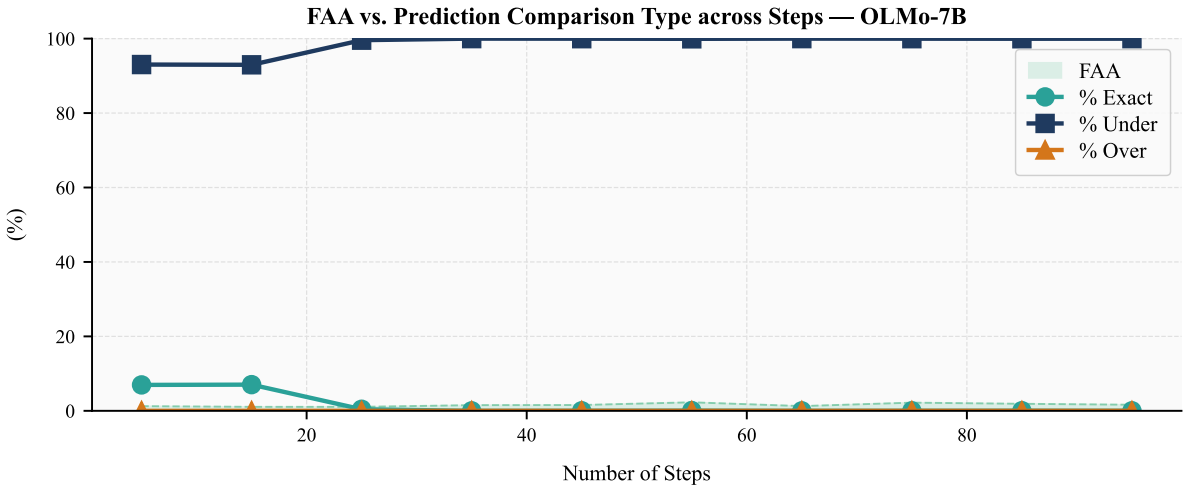


Figure 114: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). Under-execution dominates across all step lengths, with % Under remaining consistently high while % Exact stays near zero and % Over remains negligible. This indicates that the model predominantly fails to complete the required number of steps, which is associated with consistently low FAA.

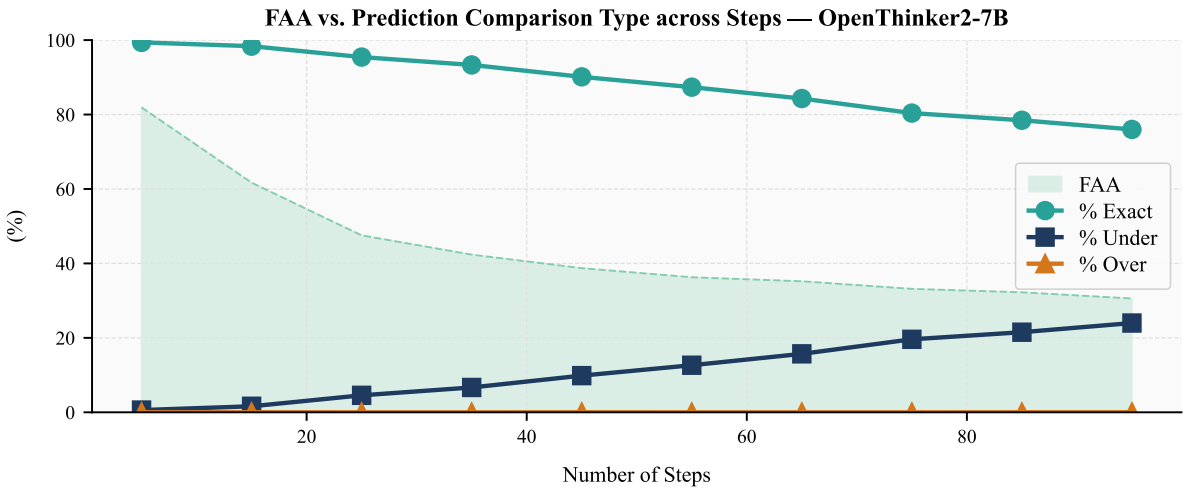


Figure 115: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). As the number of steps increases, overall FAA declines, accompanied by a slight decrease in exact-step executions and a modest increase in under-execution. These trends suggest that premature termination of step execution may contribute to the performance decline, but is unlikely to be the sole factor.

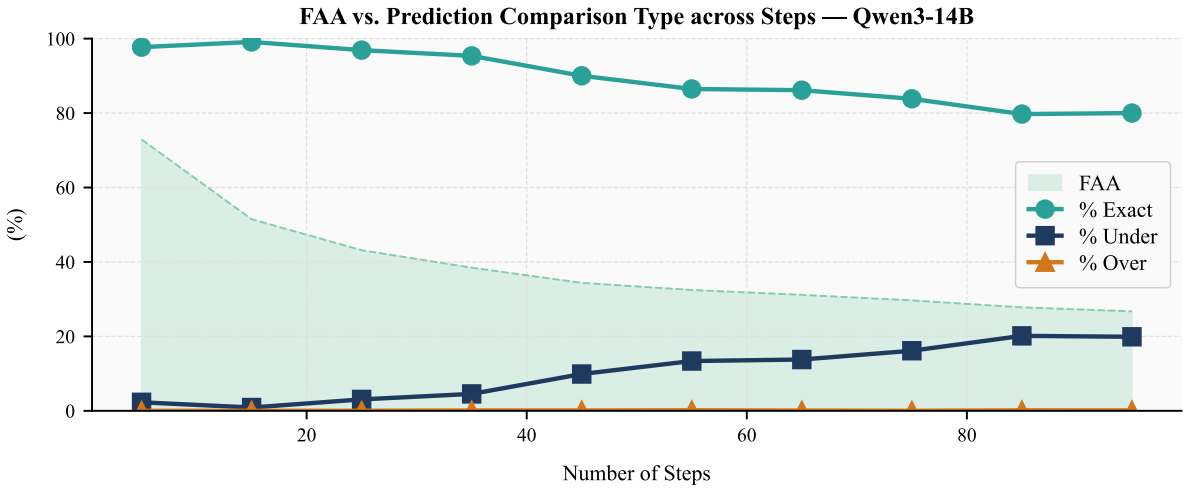


Figure 116: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). As the number of steps increases, overall FAA declines, accompanied by a slight decrease in exact-step executions and a modest increase in under-execution. These trends suggest that premature termination of step execution may contribute to the performance decline, but is unlikely to be the sole factor.

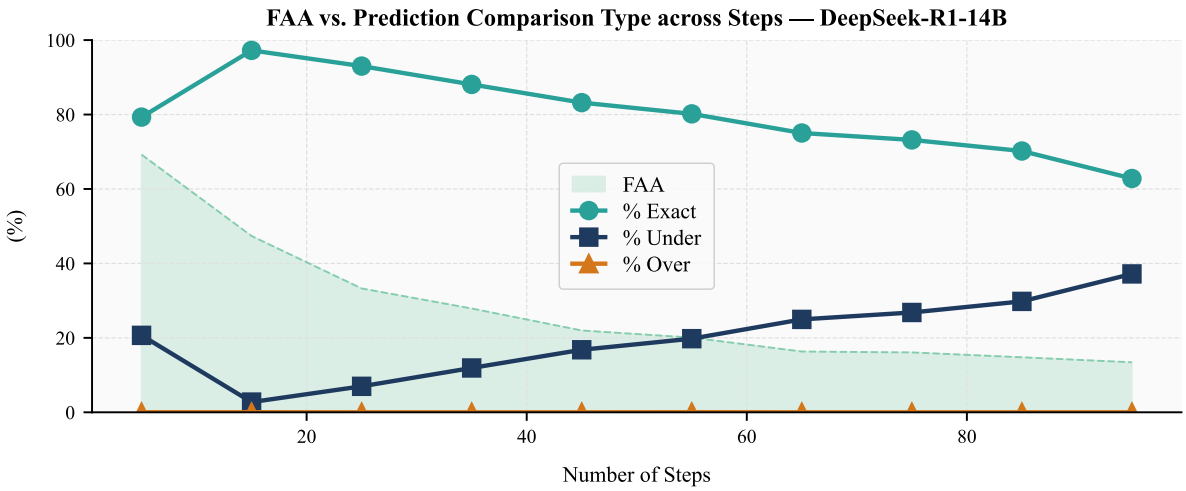


Figure 117: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). As the number of steps increases, overall FAA declines, accompanied by a slight decrease in exact-step executions and a modest increase in under-execution. These trends suggest that premature termination of step execution may contribute to the performance decline, but is unlikely to be the sole factor.

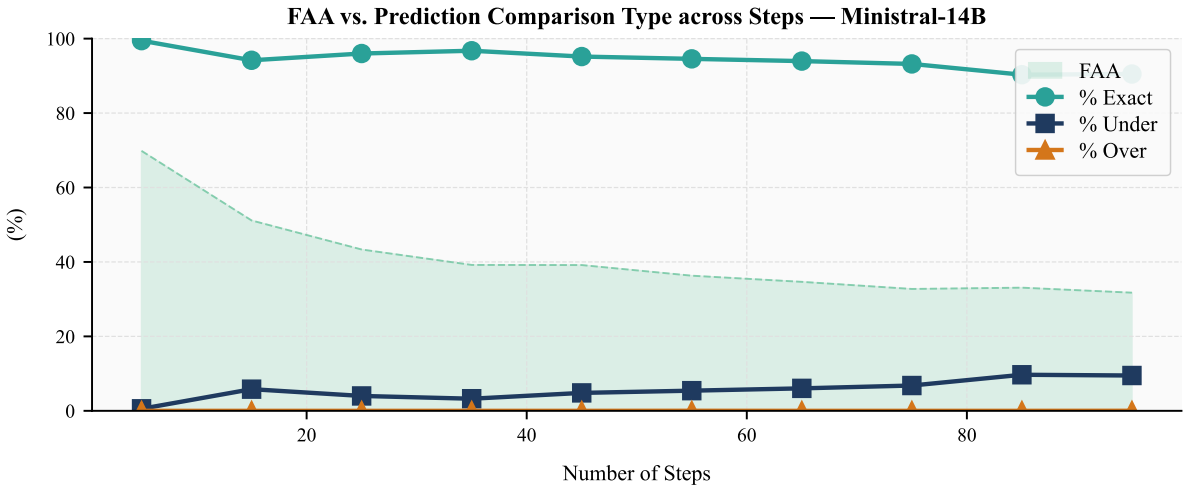


Figure 118: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). As the number of steps increases, overall FAA declines, while exact-step execution remains high with only a slight decrease, and under-execution shows a modest increase. These trends indicate that while under-execution becomes more frequent, its change is relatively small, suggesting that factors beyond premature termination also contribute to the observed performance decline.

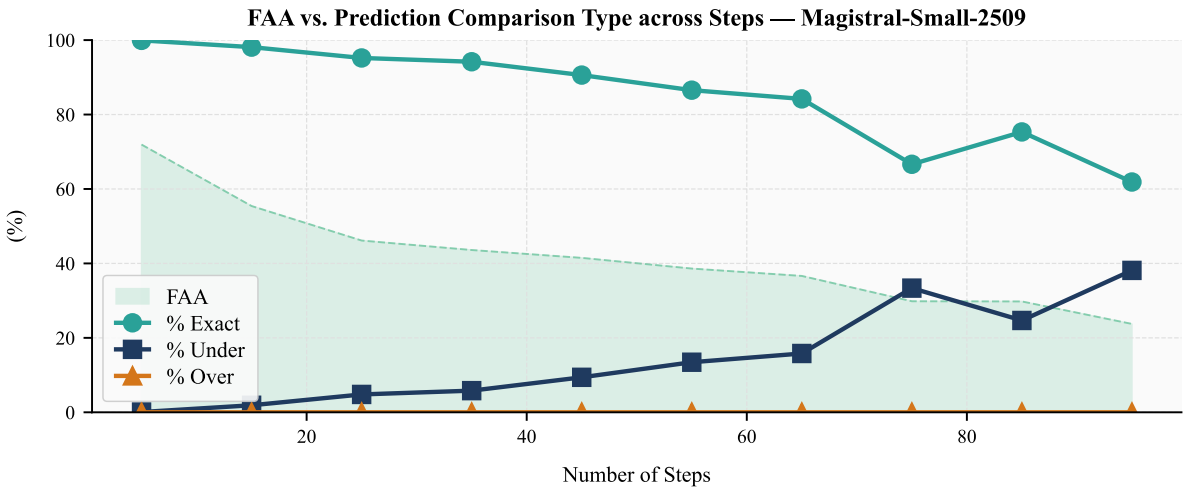


Figure 119: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). As the number of steps increases, overall FAA declines, accompanied by a slight decrease in exact-step executions and a modest increase in under-execution. These trends suggest that premature termination of step execution may contribute to the performance decline, but is unlikely to be the sole factor.

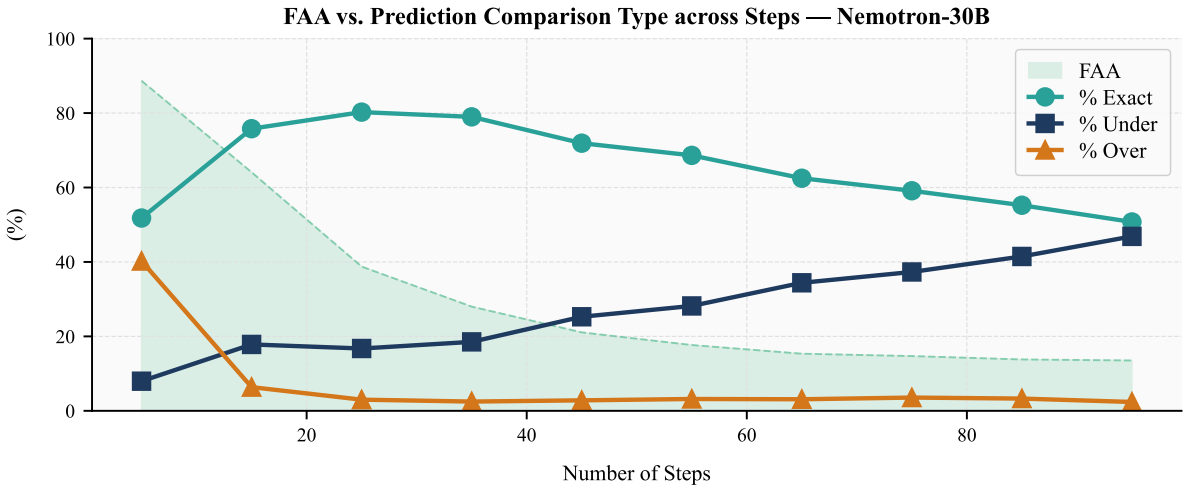


Figure 120: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). At shorter step lengths, the model exhibits relatively higher over-execution, accompanied by lower FAA. As step length increases, over-execution decreases and FAA initially improves. However, at longer step lengths, under-execution increases substantially while exact-step execution declines, coinciding with a drop in FAA. These trends suggest that both over-execution at shorter lengths and increasing under-execution at longer lengths are associated with performance degradation.

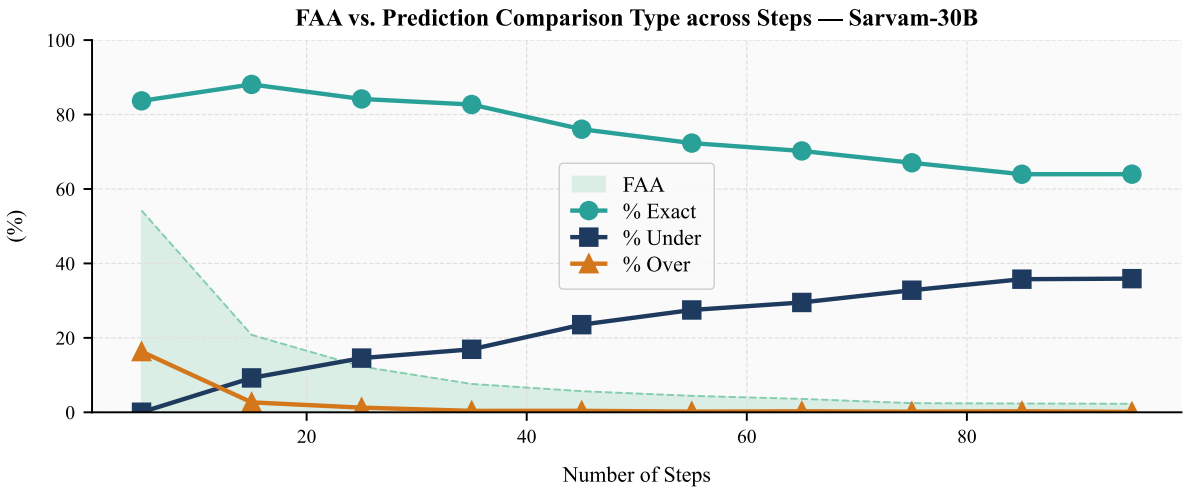


Figure 121: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). As the number of steps increases, overall FAA declines, accompanied by a slight decrease in exact-step executions and a modest increase in under-execution. These trends suggest that premature termination of step execution may contribute to the performance decline, but is unlikely to be the sole factor.

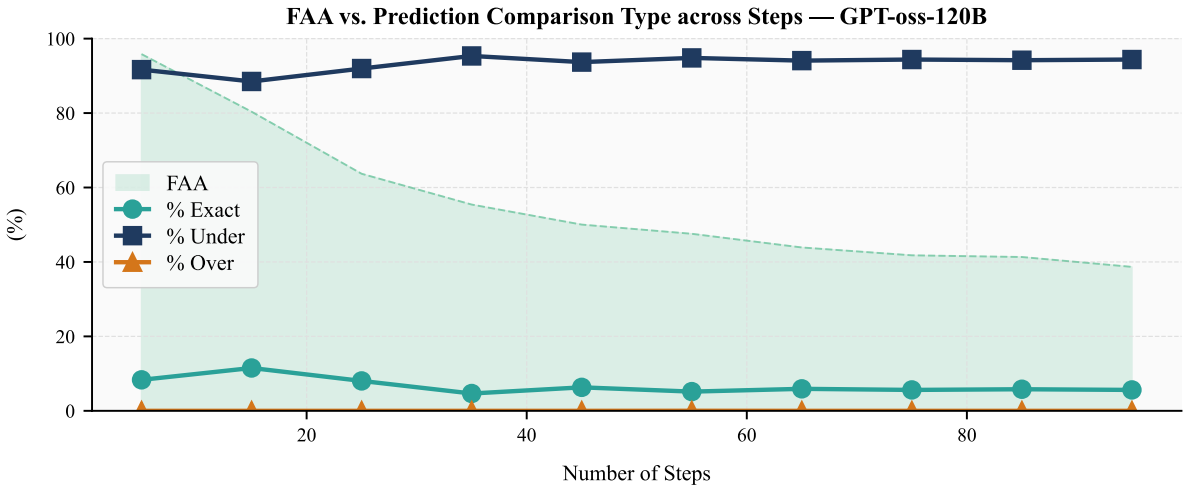


Figure 122: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). The model exhibits a strong tendency toward under-execution, indicating that it often produces outputs without completing the expected sequence of intermediate steps. In many cases, the model appears to generate the final output directly, without explicitly executing the intermediate steps.

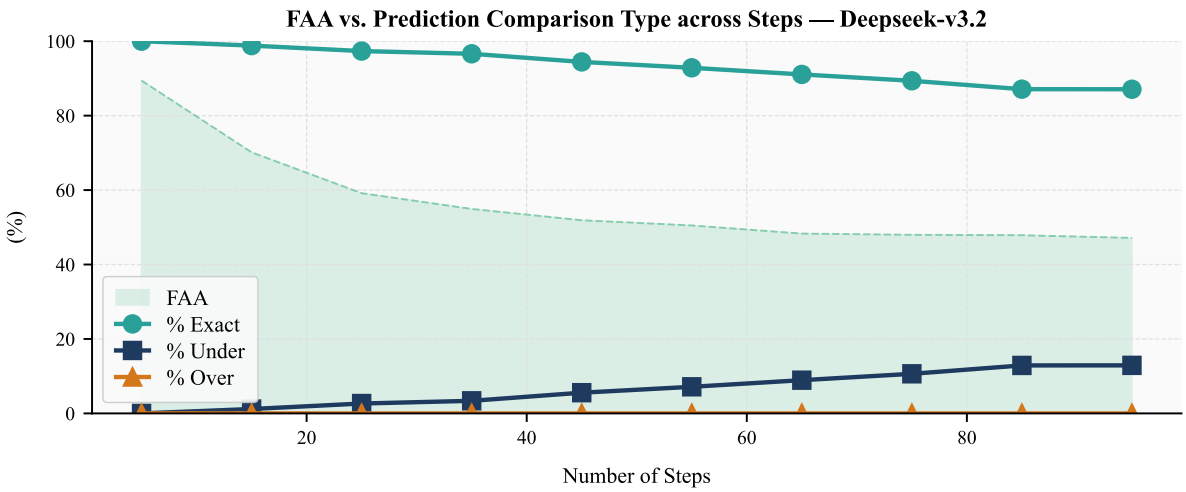


Figure 123: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). As the number of steps increases, overall FAA declines, while exact-step execution remains high with only a slight decrease, and under-execution shows a modest increase. These trends indicate that while under-execution becomes more frequent, its change is relatively small, suggesting that factors beyond premature termination also contribute to the observed performance decline.

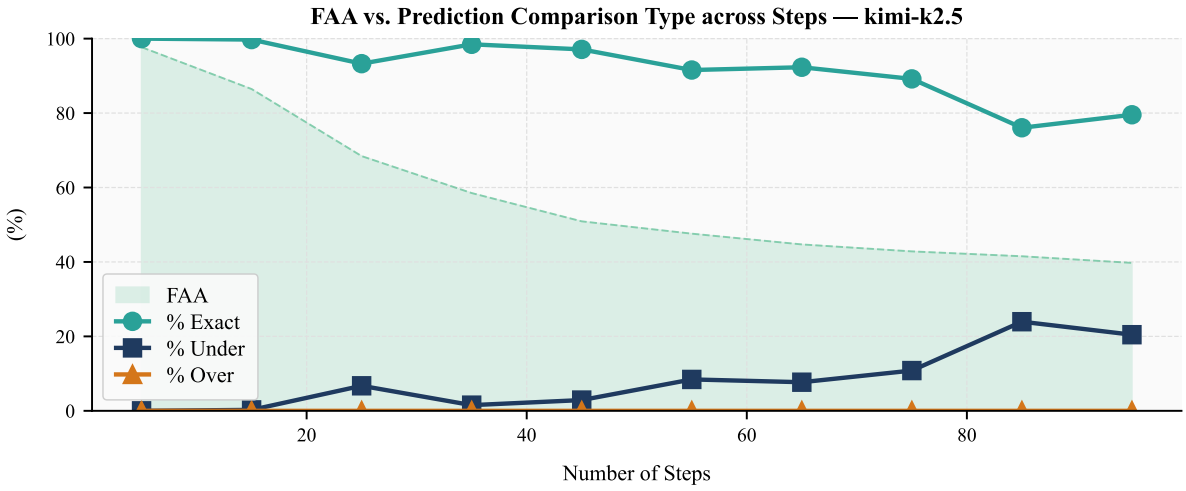


Figure 124: FAA and prediction comparison types across increasing procedure lengths. The green line (% Exact) denotes predictions where the model executes the expected number of steps, the blue line (% Under) denotes under-execution (fewer steps than expected), and the orange line (% Over) denotes over-execution (more steps than expected). As the number of steps increases, overall FAA declines, while exact-step execution remains high with only a slight decrease, and under-execution shows a modest increase. These trends indicate that while under-execution becomes more frequent, its change is relatively small, suggesting that factors beyond premature termination also contribute to the observed performance decline.

H Results: Statistical Reliability of Accuracy Estimates

Model	Accuracy (%)
Mistral-7B-Instruct-v0.3(Mistral AI, 2024)	1.51 ± 0.10
Olmo-3-7B-Think(Olmo et al., 2025)	2.00 ± 0.11
DeepSeek-R1-Distill-Qwen-1.5B(Guo et al., 2025)	3.94 ± 0.16
sarvam-30b(Sarvam Foundation Models Team, 2026)	12.07 ± 0.27
DeepSeek-R1-Distill-Qwen-7B(Guo et al., 2025)	19.71 ± 0.33
DeepSeek-R1-Distill-Qwen-14B(Guo et al., 2025)	30.98 ± 0.38
Nemotron-3-Nano30B(Blakeman et al., 2025)	33.77 ± 0.40
Qwen3-4B-Thinking-2507(Yang et al., 2025a)	38.64 ± 0.41
Qwen3-14B(Yang et al., 2025a)	42.91 ± 0.39
Ministral-3-14B-Reasoning-2512(Liu et al., 2026)	46.35 ± 0.43
Magistral-Small-2509(Rastogi et al., 2025)	47.21 ± 0.40
openthinker2(Guha et al., 2025)	48.99 ± 0.43
kimi-k2.5()	56.32 ± 0.39
deepseek-v3.2(DeepSeek-AI, 2025)	56.58 ± 0.41
gpt-oss-120b(Agarwal et al., 2025)	60.04 ± 0.42

Table 3: Model accuracy with 95% bootstrap confidence intervals. Values are reported as mean accuracy (%) $\pm$ confidence interval half-width.

Task	Median	Std. Dev.	Min	Max
Add	3.45×10^{2}	1.91×10^{3}	1.02×10^{-1}	9.56×10^{3}
Subtract	-2.89×10^{-1}	2.48×10^{2}	-1.82×10^{3}	1.36×10^{3}
Multiply	6.25×10^{31}	∞	0	1.43×10^{190}
Divide	8.75×10^{-1}	2.71×10^{33}	1.82×10^{-44}	1.40×10^{35}
Mixed	9.97×10^{-1}	7.80×10^{29}	-3.61×10^{29}	3.98×10^{31}

Table 4: Expected-output statistics across task variants. Multiplication and division exhibit substantially larger variance and heavier-tailed output distributions compared to addition and subtraction.

I Model Generations

I.1 Example 1:

Config:

```
Range: 1 to 10
Input Type: Int
Task: Mixed
Look Back: 2
Id: "4179af72-0ce8-43b8-b92a-7273e8812f27"
```

Procedure:

```
function(x, y):
Let S1 = x
Let S2 = y
Step 1: S3 = S1 + S2
Step 2: S4 = S3 - S2
Step 3: S5 = S4 - S2
Step 4: S6 = S4 - S2
Step 5: S7 = S2 + S5
Final Step: Return S7
```

Inputs:

```
x = 8
y = 7
```

Expected Output:

```
<answer>8</answer>
```

Correct Generation (OA-120):

```
<answer>8</answer>
```

Incorrect Generation (OL-7):

```
Now, execute the procedure with the given inputs.
<answer>42</answer>
```

Step-wise Correct Execution (OL-7):

```
"Let's execute the procedure step by step.
Step 1: S1 = x
S1 = 8

Step 2: S2 = y
S2 = 7

Step 3: S3 = S1 + S2
S3 = 8 + 7 = 15

Step 4: S4 = S3 - S2
S4 = 15 - 7 = 8

Step 5: S5 = S4 - S2
S5 = 8 - 7 = 1

Step 6: S6 = S4 - S2
```

```
S6 = 8 - 7 = 1

Step 7: S7 = S2 + S5
S7 = 7 + 1 = 8

Final Step: Return S7\n<answer>8</answer>"
```

Step-wise Incorrect Execution (MI-7):

```
"Output:
<answer>12</answer>

Explanation:
1. S1 = 8
2. S2 = 7
3. S3 = S1 + S2 = 8 + 7 = 15
4. S4 = S3 - S2 = 15 - 7 = 8
5. S5 = S4 - S2 = 8 - 7 = 1
6. S6 = S4 - S2 = 8 - 7 = 1 (This step is repeated, but the result is the same)
7. S7 = S2 + S5 = 7 + 1 = 8
8. Return S7 = 8

The final result is <answer>12</answer> (This is incorrect, as it
includes additional text)

Correct output:
S1 = 8
S2 = 7
S3 = 15
S4 = 8
S5 = 1
S6 = 1
S7 = 8
<answer>12</answer>"
```