seqBench: A Tunable Benchmark to Quantify Sequential Reasoning Limits of LLMs

Anonymous ACL submission

Abstract

We introduce seqBench, a parametrized benchmark for probing sequential reasoning limits in Large Language Models (LLMs) through precise, multi-dimensional control over several key complexity dimensions. seqBench allows systematic variation of (1) the *logical depth*, defined as the number of sequential actions required to solve the task; (2) the number of *backtracking* steps along the optimal path, quantifying how often the agent must revisit prior states to satisfy deferred preconditions (e.g., retrieving a key after encountering a locked door); and (3) the *noise ratio*, defined as the ratio between supporting and distracting facts about the environment. Our evaluations on state-of-the-art LLMs reveal a universal failure pattern: accuracy collapses exponentially beyond a model-specific logical depth. Unlike existing benchmarks, seqBench's fine-grained control facilitates targeted analyses of these reasoning failures, illuminating universal scaling laws and statistical limits, as detailed in this paper alongside its generation methodology and evaluation metrics. We find that even top-performing models systematically fail on seqBench's structured reasoning tasks despite minimal search complexity, underscoring key limitations in their commonsense reasoning capabilities. Designed for future evolution to keep pace with advancing models, the seqBench datasets are publicly released to spur deeper scientific inquiry into LLM reasoning, aiming to establish a clearer understanding of their true potential and current boundaries for robust real-world application.

Large Language Models (LLMs) have shown remarkable performance (Vaswani et al., 2017; Brown et al., 2020; Lieber et al., 2021; Rae et al., 2021; Smith et al., 2022; Thoppilan et al., 2022; Hoffmann et al., 2022; Du et al., 2021; Fedus et al., 2022; Zoph et al., 2022) on a wide range of tasks and benchmarks spanning diverse human-like capabilities; however, these successes can obscure fundamental limitations in sequential reasoning that still persist. Arguably, reasoning captures a more pure form of intelligence, going beyond mere pattern matching or fact memorization, and is thus a critical capability to understand and enhance in AI systems. Recent studies show that state-of-the-art LLMs (OpenAI, 2025; Google DeepMind, 2025; Meta AI, 2025; Mistral AI, 2024; Anthropic, 2025) excel at complex benchmarks, yet stumble upon simple common-sense inferences trivial for an adult human (Nezhurina et al., 2025; Han et al., 2024; Sharma, 2024; Berglund et al., 2024; Yang et al., 2019). Most existing benchmarks saturate quickly, leaving little room for fine-grained attribution studies to perform systemic probes of LLM failure modes. Consequently, a robust understanding of why and under what circumstances these models fail, especially on problems requiring sequential reasoning, remains elusive.

This gap, we argue, stems from the lack of evaluation benchmarks allowing systematic, multi-dimensional control over key independent factors that influence a task's overall reasoning difficulty. Most benchmarks (Cobbe et al., 2021; Hendrycks et al., 2021; Srivastava et al., 2023; Weston et al., 2015; Clark et al., 2018; Dua et al., 2019; Rein et al., 2023), despite their evaluation merits, often do not support a systematic variation of crucial complexity dimensions. This makes it difficult to isolate the specific conditions under which reasoning in LLMs falter. For instance, discerning whether a failure is due to the length of the required reasoning chain, the necessity to revise intermediate conclusions, or the density of distracting information is often not quantitatively possible. While prompting strategies like chain-of-thought (CoT) and model scaling have boosted aggregate performance, they often obscure sharp performance cliffs that can emerge when these underlying complexity dimensions are varied independently (Wei et al., 2023; Kojima et al., 2022). Without such

systematic control, disentangling inherent architectural limitations from those addressable via scaling (model size, data, or compute), fine-tuning, or prompting techniques is challenging. A fine-grained understanding of these performance boundaries is crucial for developing more robust and reliable reasoning systems.

To complement recent efforts (Sprague et al., 2024; Tyagi et al., 2024; Kuratov et al., 2024; Tang and Kejriwal, 2025; Mirzaee et al., 2021; Tikhonov, 2024; Mirzaee and Kordjamshidi, 2022; Shi et al., 2022) in evaluating reasoning, and to address the need for more controlled analysis, we introduce seqBench, a tunable benchmark designed explicitly to probe and analyze sequential reasoning capabilities in language models. The dataset comprises synthetic yet linguistically grounded pathfinding task configurations on two-dimensional grids. Solving each problem requires sequential inference over relevant and distracting structured facts. Each instance is automatically verifiable and parameterized by controllable factors that directly address the previously identified gaps: (1) logical depth (total number of actions in the ground-truth solution, reflecting the length of the reasoning chain); (2) backtracking count (number of locked-door detours on the optimal path, requiring revision of tentative solution paths); and (3) noise ratio (proportion of distracting vs. supporting facts, testing robustness to irrelevant information). Performance against these dimensions can be quantified with fine-grained metrics (e.g., via progress ratio as we define here). We observe that beyond a certain logical depth, Pass@1 success collapses to near zero for all models (see Figure 1). These features enable precise attribution studies of model failure modes, offering insights into the brittle boundaries of current LLM generalization.

Furthermore, the seqBench benchmark is built upon a scalable data generation framework, allowing it to evolve alongside increasingly capable models to help with both model training and evaluation. Through evaluations on popular LLMs, we reveal that top-performing LLMs exhibit steep universal declines as either of the three complexity dimensions increases, while remaining comparatively robust to fact shuffle, despite the underlying logical structure being unchanged.

Contributions. Our main contributions are:

1. **seqBench: A Tunable Benchmark for Sequential Reasoning.** We introduce an open-source framework for generating pathfinding tasks with fine-grained, orthogonal control over logical depth, backtracking steps, and noise ratio. We also evaluate secondary factors like fact ordering (shuffle ratio; See supplementary material for details).

2. **Comprehensive LLM Attribution Study.** Using seqBench, we demonstrate the significant impact of these controlled complexities on LLM performance, revealing sharp performance cliffs in state-of-the-art models even when search complexity is minimal.

The seqBench dataset is publicly available[1] under the CC BY 4.0 license to facilitate benchmarking.

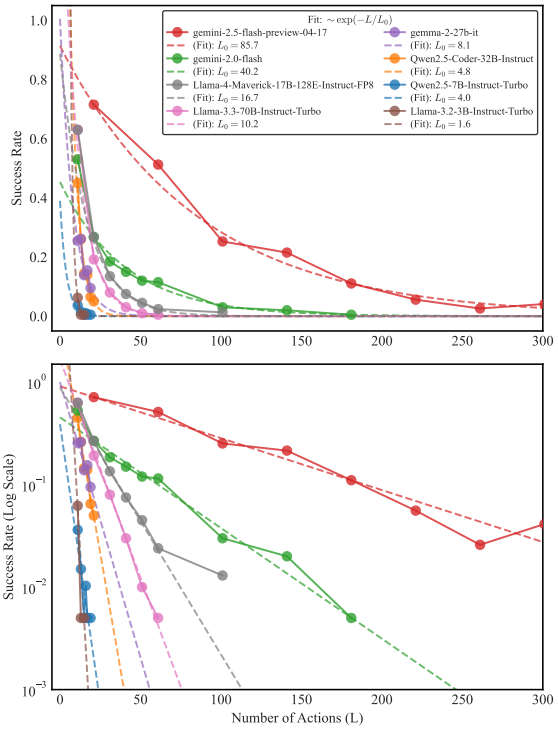

Figure 1: Performance collapse of various models with increasing logical depth L for a pathfinding task ($N, M = 40, \mathcal{B} = 2$ keys). Success rates (Pass@1) are shown on linear (top panel) and logarithmic (bottom panel) y-axes, averaged from 5 runs/problem across 40 problems per unit L-bin. All evaluations used Temperature=1.0 and top-p=0.95 (Gemini-2.5-flash: 'auto' thinking). The displayed fits employ a Weighted Least Squares (WLS) (Carroll and Ruppert, 2017) method on log-success rates. Weights are derived from inverse squared residuals of a preliminary Ordinary Least Squares (OLS) fit.

[1] https://huggingface.co/datasets/emnlp-submission/seqBench

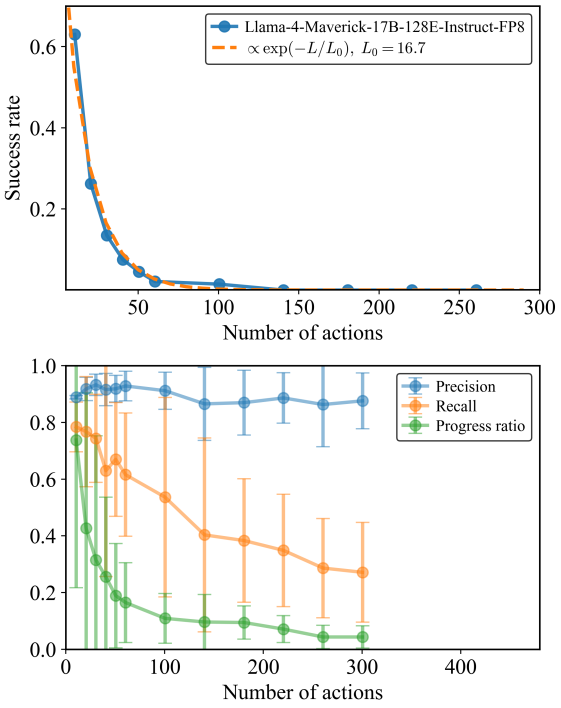

Figure 2: On the left: Llama-4 Maverick-17B-128E-Instruct Model's performance (pass@1 success rate) versus number of actions in the ground truth path of the pathfinding problems is shown. This Pass@1 success rate across 5 runs per problem is averaged over the problem instances sampled from different actions count bins of width equal to 1. On the right: The mean of progress ratio as well as precision and recall is shown to highlight models gradually increasing struggle in completing the path. The Temperature is set to 1.0 and the top-p is set to 0.95 in all runs.

1 Methods

1.1 Dataset Generation

The seqBench dataset consists of spatial pathfinding tasks. Task instance generation, detailed below (Algorithm 1; See Appendix A for details), is predicated on the precise independent control of the three key complexity dimensions introduced earlier: **Logical Depth** (L), **Backtracking Count** ($\mathcal{B}$), and **Noise Ratio** ($\mathcal{N}$). This allows the creation of instances with specific values for these parameters, enabling targeted studies of their impact on LLM reasoning.

Task instances are produced in a multi-stage process. Initially, primary generation parameters—maze dimensions (N, M), target backtracks ($\mathcal{B}_{\text{target}}$), and target noise ratio ($\mathcal{N}_{\text{target}}$)—are specified. An acyclic maze graph (M_g) is formed on an $N \times M$ grid using Kruskal's algorithm (Kleinberg and Tardos, 2006). Our "Rewind Construction" method (Algorithm 1) then embeds $\mathcal{B}_{\text{target}}$ backtracking maneuvers by working backward from a goal to strategically place keys and locked doors, yielding the instance's actual backtracking count $\mathcal{B}$. Finally, a natural language fact list ($\mathcal{F}$) is derived from the maze, and distracting facts are added according to $\mathcal{N}_{\text{target}}$ to achieve the final noise ratio $\mathcal{N}$. The *logical depth* L (optimal path length) emerges from these generative steps, influenced by $N, M, \mathcal{B}_{\text{target}}$, and construction stochasticity. While L is not a direct input to the generation algorithm, the process is designed to yield a wide spectrum of logical depths. Each generated instance is then precisely annotated with its emergent L value, alongside its effective $\mathcal{B}$ and $\mathcal{N}$ values. This annotation effectively makes L a key, selectable parameter for users of the seqBench dataset, enabling them to choose or filter tasks by their desired logical depth. Our rewind construction method guarantees task solvability. The full seqBench benchmark is constructed by systematically applying this instance generation process (detailed in Algorithm 1) across a wide range of initial parameters. This includes varied grid sizes (e.g., $N \in \{5..50\}, M \approx N$) and target backtracks ($\mathcal{B}_{\text{target}} \in \{0..7\}$), yielding a large and diverse data pool. For each $(N, M, \mathcal{B}_{\text{target}})$ configuration, multiple unique base mazes are generated, to which different noise ratios (e.g., $\mathcal{N}_{\text{target}} \in \{0..1\}$) are subsequently applied. The creation of this comprehensive data pool was computationally efficient, requiring approximately an hour of computation on a standard laptop while using minimal memory. The publicly released benchmark comprises a substantial collection of these generated instances, each annotated with its specific emergent logical depth L, effective backtracking count $\mathcal{B}$, and noise ratio $\mathcal{N}$. This rich annotation is key, enabling researchers to readily select or filter task subsets by these dimensions for targeted studies (e.g., as done for Figure 1, where instances were sampled into L-bins with other parameters fixed). For the experiments presented in this paper, specific subsets were drawn from this benchmark pool, often involving further filtering or parameter adjustments tailored to the objectives of each study; precise details for each experiment are provided in the relevant sections and figure captions. Full details on path derivation, fact compilation, and overall dataset generation parameters are provided in the Appendix A.

1.2 Prompt Construction and Model Configuration

Our evaluation uses a standardized prompt template with four components: (i) task instructions and action schema, (ii) three few-shot examples of increasing complexity (simple navigation, single-key, and multi-key backtracking), (iii) optional reasoning guidance, and (iv) the problem's natural-language facts. All models are queried using temperature T=1.0, nucleus sampling p=0.95, and maximum allowed setting in terms of output token limits on a per model basis. For each instance, we compute 5 independent runs to establish robust performance statistics. The complete prompt structure, shown in Figure 6, is provided in the Appendix B.

Algorithm 1: Rewind Construction of Path Skeleton

Input : Grid $N \times M$, Target backtracks $\mathcal{B}$
Output : Maze graph M_g, Locked doors $\mathcal{D}_L$, Key info $\mathcal{K}_I$, Path skeleton Π_S

$M_g \leftarrow$ Acyclic graph on grid (Kruskal's);
$x \leftarrow C_{goal} \leftarrow$ Random goal cell in M_g;
$\mathcal{D}_L, \mathcal{K}_I \leftarrow \emptyset, \emptyset$; $b \leftarrow 0$;
$\Pi_S \leftarrow [(C_{goal}, \text{GOAL})]$;
while $b < \mathcal{B}$ **do**
  $c_{key} \leftarrow$ Random cell in M_g accessible from x (path avoids $\mathcal{D}_L$ for this step);
  $\pi_{seg} \leftarrow$ Unique path in M_g from x to c_{key};
  if $\exists e \in \pi_{seg}$ *such that* $e \notin \mathcal{D}_L$ **then**
    $d \leftarrow$ Randomly select such an edge e;
    $\mathcal{D}_L \leftarrow \mathcal{D}_L \cup \{d\}$;
    $K_{id} \leftarrow$ New unique key ID;
    $\mathcal{K}_I[K_{id}] \leftarrow \{\text{opens} : d, \text{loc} : c_{key}\}$;
    Π_S.prepend($(c_{key}, \text{PICKUP } K_{id})$, $(d, \text{UNLOCK } K_{id})$, (π_{seg}, MOVE));
    $x \leftarrow c_{key}$; $b \leftarrow b + 1$;
  end
  else
    Break
  end
end
Π_S.prepend((x, START));
return $M_g, \mathcal{D}_L, \mathcal{K}_I, \Pi_S$;

1.3 Evaluation Metrics

To analyze not just success but also *how* models fail, we employ several complementary metrics. **Success Rate (Pass@1)** measures the proportion of runs where the predicted action sequence exactly matches the ground truth. The **Progress Ratio** (Tyagi et al., 2024), calculated as k/n (where n is the total ground-truth actions and k is the number correctly executed before the first error), pinpoints the breakdown position in reasoning. We also use **Precision** and **Recall**. Precision is the proportion of predicted actions that are correct, while Recall is the proportion of ground-truth actions that were correctly predicted. Low precision indicates hallucinated actions, while low recall signifies missed necessary actions. Additionally, we visualize error locations via a **Violation Map**. This multi-faceted approach reveals each model's effective "reasoning horizon"—the maximum sequence length it can reliably traverse. Further details on all metrics and visualizations are provided in the supplementary material.

2 Benchmarking Results

2.1 Evaluated Models

We evaluate a diverse set of transformer-based LLMs across different model families and parameter scales. Our analysis includes Gemini models (2.5-flash-preview, 2.0-flash), Meta's Llama family (4-Maverick-17B, 3.3-70B, 3.2-3B), Google's Gemma-2-27b, and Qwen models (2.5-Coder-32B, 2.5-7B). Access to some open-weight models and benchmarking infrastructure was facilitated by platforms such as Together AI[2] and Google AI Studio[3]. Problem instances for varying logical depths (L) were generated by sampling 40 problems for each L, using a fixed maze size of 40×40 and 2 keys, unless otherwise specified for specific experiments (e.g., when varying the number of keys for backtracking analysis). All models were evaluated using the standardized prompt template (see Figure 6), the inference settings detailed in Section 1.2, and a common response parsing methodology. For each task instance, we perform 5 independent runs to establish robust performance statistics, primarily analyzing Pass@1 success rates.

2.2 Universal Performance Collapse with Increasing Logical Depth

A central finding of our study is the universal collapse in reasoning performance observed across all evaluated LLMs when confronted with tasks requiring increasing sequential inference steps. As

[2] https://www.together.ai/
[3] https://aistudio.google.com/

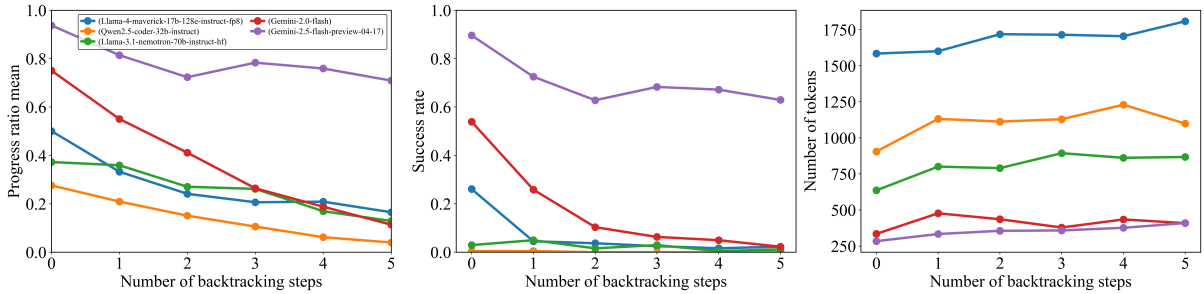

Figure 3: Performance as a function of the number of required backtracking steps, operationalized via the number of locked doors with distributed keys along the optimal path. Holding all other complexity factors constant, all models exhibit a clear decline in both progress ratio and success rate as backtracking demands increase. Additionally, we report the corresponding rise in output token counts per model, highlighting the increased reasoning burden associated with longer dependency chains.

illustrated in Figure 1, Pass@1 success rates exhibit a consistent and sharp exponential decay as the ground-truth path length (L) increases. Performance rapidly approaches near-zero past a model-specific point in this decay. To quantify and compare this exponential decay, we fit an exponential decay curve $P(L) = \exp(-L/L_0)$ to the success rates, deriving a characteristic path length L_0. This L_0 value, representing the path length at which performance drops by a factor of e^{-1}, serves as a robust metric for each model's sequential reasoning horizon. Plotting success rates on a semi-logarithmic (log-y) scale against L reveals an approximately linear decay trend across the evaluated regime. This log-linear relationship suggests that errors may accumulate with a degree of independence at each reasoning step, eventually overwhelming the model's capacity for coherent inference. The observed L_0 values vary significantly, from 85.7 for Gemini-2.5-Flash down to 1.6 for Llama-3.2-3B (Figure 1), underscoring a fundamental bottleneck in current transformer architectures for extended multi-step reasoning.

2.3 Impact of Independently Controlled Complexity Dimensions

Beyond the universal impact of logical depth (L) discussed in Section 2.2, our benchmark's ability to independently vary key complexity dimensions allows for targeted analysis of their distinct impacts on LLM reasoning performance. We highlight the effects of noise, backtracking, and fact ordering, primarily focusing on Pass@1 success rates, mean progress ratios, and response token counts.

Impact of Backtracking Requirements. Increasing the number of required backtracking steps—operationalized via key-door mechanisms—also leads to a clear and significant decline in Pass@1 success rates and mean progress ratios across all evaluated models (Llama-4 Maverick, Qwen 2.5-Code, Llama-3.2 Nemotron, Gemini 2.0 Flash, and Gemini 2.5 Flash-preview), as shown in Figure 3. Gemini 2.5 Flash-preview maintains the highest performance but still exhibits a notable drop as backtracking count increases from 0 to 5. This decline in reasoning accuracy is generally accompanied by an increase or sustained high level in the mean number of response tokens (Figure 3, right panel). For example, models like Llama-4 Maverick and Gemini 2.5 Flash-preview show a clear upward trend or maintain high token counts as backtracking complexity rises, reflecting the increased reasoning effort or path length articulated by the models when managing more complex sequential dependencies.

Sensitivity to Noise Ratio. Model performance is highly sensitive to the noise ratio—the proportion of distracting versus supporting facts. As demonstrated in Figure 4 for Gemini 2.5 Flash and Llama-4 Maverick, increasing the proportion of irrelevant facts consistently and significantly degrades both Pass@1 success rates and mean progress ratios. For instance, Gemini 2.5 Flash's Pass@1 success rate drops from over 0.7 at zero noise to approximately 0.2 at a noise ratio of 1.0. Llama-4 Maverick, starting with lower performance, also shows a consistent decline. Interestingly, for these two models, the number of CoT (output) tokens remains relatively stable despite the increasing noise and degrading performance (Figure 4, right panel), suggesting that models do not necessarily "work harder" (in terms of output

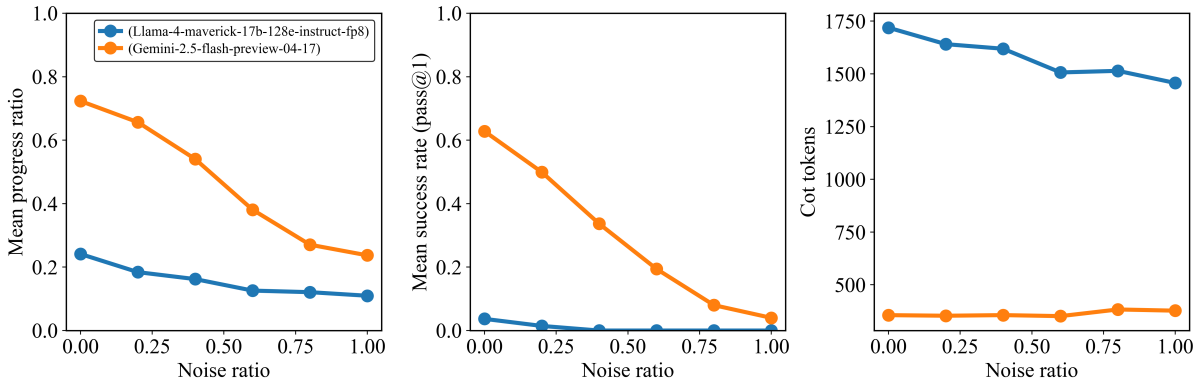

Figure 4: Performance as a function of contextual noise for Gemini 2.5 flash and Llama-4 Maverick-17B-128E-Instruct models. As noise increases through the inclusion of distracting or irrelevant facts, both models exhibit a clear and consistent decline in performance.

length) when faced with more distractors, but their accuracy suffers.

Fact Ordering (Shuffle Ratio). In contrast to the strong effects of noise and backtracking, shuffle ratio (entropy of fact presentation order) within the prompt appears to play a secondary role when varied in isolation. Our experiments, exemplified by the performance of Gemini 2.5 Flash and Llama-4 Maverick (see Appendix C Figure 14 for details), show that complete shuffling of facts (randomizing their presentation order without adding or removing any information) has a minimal impact on Pass@1 success rates and mean progress ratios. Output token counts also remain stable. This suggests a relative robustness to presentation order as long as all necessary information is present and distinguishable. However, as details provided in supplementary material, when high noise and high shuffle co-occur, the combined effect can be more detrimental than either factor alone, though noise remains the dominant degrading factor.

2.4 Characterizing Key Failure Modes and Error Patterns

A Key Failure Mode: Omission of Critical Steps. Beyond simply taking illegal shortcuts, detailed analysis reveals that LLMs often fail by omitting critical sub-goals necessary for task completion. Figure 2 (bottom panel) provides a quantitative view for Llama-4 Maverick (Meta AI, 2025), showing that while precision generally remains high (models infrequently hallucinate non-existent rooms or facts), recall and progress ratio plummet with increasing path length (L). This indicates that models predominantly fail by missing necessary actions or entire crucial sub-sequences. For a qualitative example, even capable models like Gemini-2.5-Flash can neglect essential detours, such as collecting a required key, thereby violating sequential dependencies and rendering the task unsolvable (illustrative examples are provided in the Appendix B.4; see Figures 8 and 9). This pattern highlights a fundamental breakdown in robust multi-step planning and execution.

Path-Length Dependent First Errors: The Burden of Anticipated Complexity. The propensity for models to make critical errors is not uniformly distributed across the reasoning process, nor is it solely a feature of late-stage reasoning fatigue. Examining the distribution of steps at which the first constraint violations occur reveals a counterintuitive pattern: as the total required path length (L) of a problem increases, models tend to fail more frequently *even at the earliest steps* of the reasoning chain. This leftward shift in the first-error distribution also observed under increasing noise, (Appendix B.4; Figures 10 and 11) contradicts a simple cumulative error model where each step carries a fixed, independent failure probability. Instead, an error at an early step (e.g., step 5) becomes substantially more likely when the model is attempting to solve an 80-step problem versus a 20-step problem. This suggests that the overall anticipated complexity of the full problem influences reasoning quality from the very outset, indicating a struggle with global planning or maintaining coherence over longer horizons, rather than just an accumulation of local errors. This phenomenon may help explain why prompting techniques that decompose long problems into smaller, manageable sub-problems

often succeed.

2.5 Disparity: Information Retention vs. Reasoning Capacity

On seqBench tasks, this disparity is quantitatively striking. While modern LLMs boast million-token contexts, their effective sequential reasoning depth typically remains on the order of hundreds of actions (Figure 1). This functional limit, even at several hundred actions (e.g., 300 actions, with each like `('move_to', 'A12')` being 5-7 tokens, totaling 1.5k-2.1k tokens), still consumes a minute fraction of their nominal context. Consequently, the ratio of context capacity to reasoning tokens often spans from several hundred-fold (e.g., 500:1 for 300 actions consuming 2k tokens within a 1M context) to potentially higher values given fewer limiting actions or larger model contexts. This striking gap suggests that while transformers can store and retrieve vast information, their ability to reliably chain it for coherent, multi-step inference appears surprisingly constrained.

2.6 Challenging the Conventional Performance Hierarchy

While metrics like average L_0 provide a general ranking of model capabilities, our fine-grained analysis reveals instances that challenge a simple linear performance hierarchy. Scatter plots of progress ratios across different models on identical tasks (see Appendix C Figure 13) show intriguing cases where models with lower overall L_0 values (i.e., typically weaker models) occasionally solve specific complex problems perfectly, while models with higher average L_0 values fail on those same instances. These performance inversions suggest that sequential reasoning failures may not solely stem from insufficient scale (parameters or general training) but could also arise from more nuanced reasoning limitations.

3 Related Work

Recent advancements in benchmarks evaluating sequential reasoning capabilities of LLMs have illuminated various strengths and limitations across different dimensions of complexity. These benchmarks typically differ in how they isolate and quantify reasoning challenges, such as logical deduction, retrieval difficulty, combinatorial complexity, and sensitivity to irrelevant information. ZebraLogic (Lin et al., 2025), for instance, targets formal deductive inference through logic-grid puzzles framed as constraint-satisfaction problems (csp, 2008). While valuable for probing deduction, its core methodology leads to a search space that grows factorially with puzzle size (Sempolinski, 2009). This makes it challenging to disentangle intrinsic reasoning failures from the sheer combinatorial complexity of the search. As the ZebraLogic authors themselves acknowledge: "*solving ZebraLogic puzzles for large instances may become intractable... the required number of reasoning tokens may increase exponentially with the size of the puzzle.*" This inherent characteristic means that for larger puzzles, performance is primarily dictated by the manageability of the search space rather than the limits of sequential reasoning depth. GridPuzzle (Tyagi et al., 2024) complements this by providing a detailed error taxonomy for grid puzzles, focusing on *what* kinds of reasoning mistakes LLMs make. However, like ZebraLogic, it doesn't offer independent control over key complexity dimensions such as logical depth, backtracking needs, or noise, separate from the puzzle's inherent search complexity.

Other benchmarks conflate reasoning with different cognitive demands. BABILong (Kuratov et al., 2024) tests models on extremely long contexts (up to 50M tokens), primarily assessing the ability to retrieve "needles" (facts) from a "haystack" (distracting text that does not contribute to solving the task). While valuable for evaluating long-context processing, this design makes it hard to disentangle retrieval failures from reasoning breakdowns, as performance is often dictated by finding the relevant information rather than reasoning over it. MuSR (Sprague et al., 2024) embeds reasoning tasks within lengthy narratives (e.g., murder mysteries), mixing information extraction challenges with complex, domain-specific reasoning structures. This realism obscures which specific aspect—extraction or reasoning depth—causes model failures. DynabAbI (Tamari et al., 2021) offers a dynamic framework for compositional generalization but focuses on qualitative combinations rather than systematically varying quantitative complexity metrics needed to find precise failure points.

Spatial reasoning benchmarks, while relevant, also target different aspects. GRASP (Tang and Kejriwal, 2025) assesses practical spatial planning efficiency (like obstacle avoidance) in 2D grids, a different skill than the abstract sequential reasoning seqBench isolates. SPARTQA (Mirzaee et al., 2021) focuses on specialized spatial relational complexity (transitivity, symmetry) using

coupled dimensions, preventing independent analysis of factors like path length. SpaRTUN (Mirzaee and Kordjamshidi, 2022) uses synthetic data primarily for transfer learning in Spatial Question Answering (SQA), aiming to improve model performance rather than serve as a diagnostic tool with controllable complexity. Similarly, StepGame (Shi et al., 2022) demonstrates performance decay with more reasoning steps in SQA but lacks the fine-grained, orthogonal controls over distinct complexity factors provided by seqBench.

In contrast, seqBench takes a targeted diagnostic approach. By deliberately simplifying the spatial environment to minimize search complexity, it isolates sequential reasoning. Its core contribution lies in the independent, fine-grained control over (1) logical depth (the number of sequential actions required to solve the task), (2) backtracking count (the number of backtracking steps along the optimal path), and (3) noise ratio (the ratio of supporting to distracting facts). This orthogonal parameterization allows us to precisely pinpoint *when* and *why* sequential reasoning capabilities degrade, revealing fundamental performance cliffs even when search and retrieval demands are trivial. seqBench thus offers a complementary tool for understanding the specific limitations of sequential inference in LLMs.

4 Conclusion

We introduced seqBench, a novel benchmark framework designed for the precise attribution of sequential reasoning failures in Large Language Models. seqBench's core strength lies in its unique capability for fine-grained, independent control over fundamental complexity dimensions; most notably, logical depth (L), backtracking requirements, and noise ratio, its provision of automatically verifiable solutions, and critically minimizing confounding factors like search complexity. This design allows seqBench to isolate and rigorously evaluate the sequential inference capabilities of LLMs, enabling the automatic quantification of fine-grained performance metrics (such as progress ratio) and providing a clear lens into mechanisms often obscured in most other benchmarks. The framework's inherent scalability and open-source nature position it as a durable tool for assessing and driving progress in current and future generations of models, ultimately aiming to enhance their utility for complex, real-world problems that often span multiple domains. Our comprehensive evaluations using seqBench reveal that reasoning accuracy consistently collapses exponentially with increasing logical depth across a diverse range of state-of-the-art LLMs. This collapse is characterized by a model-specific parameter L_0 (Section 2.2), indicating an inherent architectural bottleneck in maintaining coherent multi-step inference. In alignment with the goal of advancing NLP's reach and fostering its responsible application in other fields by offering this precise analysis, seqBench provides a valuable resource. It encourages a shift beyond aggregate benchmark scores towards a more nuanced understanding of model capabilities, an essential step for rigorously assessing the true impact and potential risks of applying LLMs in new domains. The insights gleaned from seqBench can inform both NLP developers in building more robust models, and experts in other disciplines in setting realistic expectations and co-designing NLP solutions that are genuinely fit for purpose. Targeted improvements, guided by such fundamental understanding, are key to enhancing the robustness of sequential reasoning, making LLMs more reliable partners in interdisciplinary endeavors. Future work should leverage these insights to develop models that can overcome the observed performance cliffs and extend their effective reasoning horizons, thereby unlocking their transformative potential in diverse interdisciplinary applications—such as navigating complex scientific literature, supporting intricate legal analysis, or enabling robust multi-step planning in critical autonomous systems. Focusing on commonsense reasoning is paramount for NLP to achieve transformative societal impact, moving beyond incremental improvements to genuine breakthroughs.

5 Limitations

While seqBench offers precise control over key reasoning complexities, our study has limitations that open avenues for future research:

1. **Generalizability and Task Design Fidelity:** Our current findings are rooted in synthetic spatial pathfinding tasks. While this allows for controlled experimentation, future work must extend seqBench's methodology to more diverse reasoning domains (e.g., mathematical proofs) and incorporate greater linguistic diversity (e.g., ambiguity) to assess the broader applicability of the observed phenomena of

performance collapse (quantified by L_0) and failure patterns.

2. **Model Scope and Understanding Deeper Failure Dynamics**: Our current evaluation, while covering diverse public models, should be expanded to a wider array of LLMs—including recent proprietary and newer open-source variants (e.g., GPT, Claude, DeepSeek series)—to rigorously assess the universality of our findings on the characteristic length L_0 and failure patterns. Furthermore, while `seqBench` effectively characterizes how reasoning performance degrades with logical depth (i.e., by determining L_0), two complementary research thrusts are crucial for understanding *why*. First, systematic investigation is needed to disentangle how L_0 is influenced by factors such as model architecture, scale (parameters, training data, compute), fine-tuning strategies, and inference-time computation (e.g., chain-of-thought depth). Second, deeper analysis is required to explain the precise mechanisms underlying the observed exponential performance collapse characterized by L_0 and to account for other non-trivial error patterns, such as path-length dependent first errors.

3. **Impact of Prompting:** Our current study employed standardized prompts and inference settings. A crucial next step is a robust sensitivity analysis to determine overall decay behavior are influenced by different prompting strategies (e.g., zero-shot vs. few-shot, decomposition techniques), varied decoding parameters (temperature, top-p), and interactive mechanisms such as self-verification or self-correction. Investigating the potential of these techniques to mitigate the observed sequential inference failures, particularly given `seqBench`'s minimal search complexity, remains a key avenue for future research.

Addressing these points by leveraging frameworks like `seqBench` will be vital for developing LLMs with more robust and generalizable sequential reasoning capabilities, and for understanding their fundamental performance limits.

References

2008. Rina dechter , constraint processing, morgan kaufmann publisher (2003) isbn 1-55860-890-7, francesca rossi, peter van beek and toby walsh, editors, handbook of constraint programming, elsevier (2006) isbn 978-0-444-52726-4. *Computer Science Review*, 2:123–130.

Anthropic. 2025. Claude 3.7 sonnet. https://www.anthropic.com/news/claude-3-7-sonnet.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2024. The reversal curse: Llms trained on "a is b" fail to learn "b is a". *Preprint*, arXiv:2309.12288.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Raymond J Carroll and David Ruppert. 2017. *Transformation and weighting in regression*. Chapman and Hall/CRC.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

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

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, and 8 others. 2021. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *Preprint*, arXiv:1903.00161.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Google DeepMind. 2025. Gemini 2.5 pro experimental. https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/.

Pengrui Han, Peiyang Song, Haofei Yu, and Jiaxuan You. 2024. In-context learning may not elicit trustworthy reasoning: A-not-b errors in pretrained language models. *Preprint*, arXiv:2409.15454.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. Training compute-optimal large language models. *Preprint*, arXiv:2203.15556.

Jon Kleinberg and Eva Tardos. 2006. *Algorithm Design*. Pearson/Addison-Wesley, Boston.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Yury Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554.

Opher Lieber, Or Sharir, Barak Lenz, and Yoav Shoham. 2021. Jurassic-1: Technical details and evaluation. https://www.ai21.com/blog/jurassic-1-technical-details-and-evaluation. White Paper.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. 2025. Zebralogic: On the scaling limits of llms for logical reasoning. *Preprint*, arXiv:2502.01100.

Meta AI. 2025. Llama 4: Open and efficient multimodal language models. https://github.com/meta-llama/llama-models.

Roshanak Mirzaee, Hossein Rajaby Faghihi, Qiang Ning, and Parisa Kordjmashidi. 2021. Spartqa: : A textual question answering benchmark for spatial reasoning. *Preprint*, arXiv:2104.05832.

Roshanak Mirzaee and Parisa Kordjamshidi. 2022. Transfer learning with synthetic corpora for spatial role labeling and reasoning. *Preprint*, arXiv:2210.16952.

Mistral AI. 2024. Mistral large 2. https://mistral.ai/news/mistral-large-2407.

Marianna Nezhurina, Lucia Cipolina-Kun, Mehdi Cherti, and Jenia Jitsev. 2025. Alice in wonderland: Simple tasks showing complete reasoning breakdown in state-of-the-art large language models. *Preprint*, arXiv:2406.02061.

OpenAI. 2025. Openai o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Matthias Rauh, Po-Sen Huang, and 58 others. 2021. Scaling language models: Methods, analysis & insights from training Gopher. *Preprint*, arXiv:2112.11446.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof qa benchmark. *Preprint*, arXiv:2311.12022.

Peter Sempolinski. 2009. Automatic solutions of logic puzzles.

Manasi Sharma. 2024. Exploring and improving the spatial reasoning abilities of large language models. In *I Can't Believe It's Not Better Workshop: Failure Modes in the Age of Foundation Models*.

Zhengxiang Shi, Qiang Zhang, and Aldo Lipani. 2022. Stepgame: A new benchmark for robust multi-hop spatial reasoning in texts. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 11321–11329.

Samuel Smith, Mostofa Patwary, Brian Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhenhao Liu, Shrimai Prabhumoye, Georgios Zerveas, Vikas Korthikanti, Eric Zhang, Rewon Child, Reza Yazdani Aminabadi, Jared Bernauer, Xia Song Song, Mohammad Shoeybi, Yuxin He, Michael Houston, Shishir Tiwary, and Bryan Catanzaro. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *Preprint*, arXiv:2201.11990.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2024. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. *Preprint*, arXiv:2310.16049.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, and 432 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Preprint*, arXiv:2206.04615.

Ronen Tamari, Kyle Richardson, Aviad Sar-Shalom, Noam Kahlon, Nelson Liu, Reut Tsarfaty, and Dafna Shahaf. 2021. Dyna-babi: unlocking babi's potential with dynamic synthetic benchmarking. *Preprint*, arXiv:2112.00086.

Zhisheng Tang and Mayank Kejriwal. 2025. Grasp: A grid-based benchmark for evaluating commonsense spatial reasoning. *Preprint*, arXiv:2407.01892.

Rami Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yi Du, Yanping Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Max Krikun, Dmitry Lepikhin, James Qin, and 38 others. 2022. Lamda: Language models for dialog applications. arXiv preprint. Technical report, Google Research.

Alexey Tikhonov. 2024. Plugh: A benchmark for spatial understanding and reasoning in large language models. *Preprint*, arXiv:2408.04648.

Nemika Tyagi, Mihir Parmar, Mohith Kulkarni, Aswin RRV, Nisarg Patel, Mutsumi Nakamura, Arindam Mitra, and Chitta Baral. 2024. Step-by-step reasoning to solve grid puzzles: Where do llms falter? *Preprint*, arXiv:2407.14790.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

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.

Kaiyu Yang, Olga Russakovsky, and Jia Deng. 2019. SpatialSense: An adversarially crowdsourced benchmark for spatial relation recognition. In *International Conference on Computer Vision (ICCV)*.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models. *Preprint*, arXiv:2202.08906.

Appendices

A Dataset Generation Details

The seqBench benchmark generates pathfinding tasks by systematically controlling several complexity dimensions. As described in Section 1 (main paper), Algorithm 1 is central to this process. This appendix provides further details on the generation phases, natural language encoding of tasks, and specific dataset parameters.

A.1 Generation Phases

The generation process, guided by Algorithm 1, involves three main phases:

1. **Base Maze Construction:** An initial $N \times M$ grid is populated, and an acyclic maze graph (M_g) is formed using Kruskal's algorithm (Kleinberg and Tardos, 2006). This ensures a simply connected environment where a unique path exists between any two cells if all internal "walls" (potential door locations) were open. The overall process results in maze instances like the one visualized in Figure 5.

2. **Rewind Construction for Path Skeleton and Key/Door Placement:** This phase implements the "Rewind Construction" (Algorithm 1 in the main paper). Starting from a randomly selected goal cell (C_{goal}), the algorithm works backward to define a solvable path skeleton (Π_S). It iteratively:
   (a) Selects a cell c_{key} that would be a preceding point on a path towards the current cell x (initially C_{goal}).
   (b) Identifies the unique path segment π_{seg} in M_g from x to c_{key}.
   (c) Randomly selects an edge d on this segment π_{seg} to become a locked door. This edge d is added to the set of locked doors $\mathcal{D}_L$.
   (d) A new unique key K_{id} is conceptually placed at c_{key}, and its information (which door it opens, its location) is stored in $\mathcal{K}_I$.
   (e) The conceptual steps (moving along π_{seg}, unlocking door d with K_{id}, picking up K_{id} at c_{key}) are prepended (in reverse logical order) to the path skeleton Π_S.
   (f) The current cell x is updated to c_{key}, and the process repeats until the target number of backtracks ($\mathcal{B}$) is achieved or no valid placements remain.

   This backward construction ensures solvability and controlled backtracking complexity. The final agent starting position is the cell x at the end of this phase.

3. **Fact Compilation and Noise Injection:** Based on the final maze structure $(M_g, \mathcal{D}_L, \mathcal{K}_I)$, a set of natural language facts $\mathcal{F}$ is compiled. This includes facts describing room connections, key locations, and door states. Distracting facts are then introduced based on the target noise ratio $\mathcal{N}$. These distractors might describe non-existent connections, spurious keys, or misleading adjacencies, chosen to be plausible yet incorrect.

A.2 Natural Language Encoding

Each task instance is translated into a set of atomic natural language facts. We use a consistent templating approach:

- **Room Connections:** "Room A1 and B1 are connected by an open door."
- **Locked Connections:** "Room C3 and D3 are connected by a closed and locked door."
- **Key Requirements:** "The locked door between C3 and D3 requires key 5." (Key IDs are simple integers).
- **Key Placements:** "Key 5 is in room E4." (Room IDs use spreadsheet-like notation, e.g., A1, B2).
- **Starting Position:** "Bob is in room A2."
- **Goal Position:** "Alice is in room D5."

The full set of facts for a given problem constitutes its description.

A.3 Dataset Parameters and Scope

The seqBench dataset was generated using the following parameter ranges based on the generation configuration:

- **Grid Sizes ($N \times M$):** $N \times M$ where N and M range from 5 to 50 (e.g., [5,5], [6,6], ..., [50,50]), with $M = N$ for all configurations.

- **Target Backtracking Steps ($\mathcal{B}$):** Values from 0 to 7. This controls the number of key-door mechanisms deliberately placed on the optimal path.

- **Noise Ratio ($\mathcal{N}$):** Values from 0.0 (no distracting facts) to 1.0 (equal number of supporting and distracting facts), typically in increments of 0.2.

- **Instances per Configuration:** For each primary configuration, defined by a specific grid size (N, M) and a specific target backtracking step count ($\mathcal{B} \in \{0..7\}$), 400 unique base maze instances were generated.

- **Logical Depth (L):** As an emergent property, L varies. Experiments typically select problems from these generated instances that fall into specific L bins (e.g., $L \in [10, 11), [11, 12), \ldots$).

This generation pipeline, leveraging the described parameter ranges and variations, can produce a vast and diverse set of problem instances. The publicly released `seqBench` dataset, used for the analyses in this paper (see main paper for access link), comprises 7,079 such curated instances. This collection offers a rich resource for studying the combined effects of the controlled complexity dimensions.

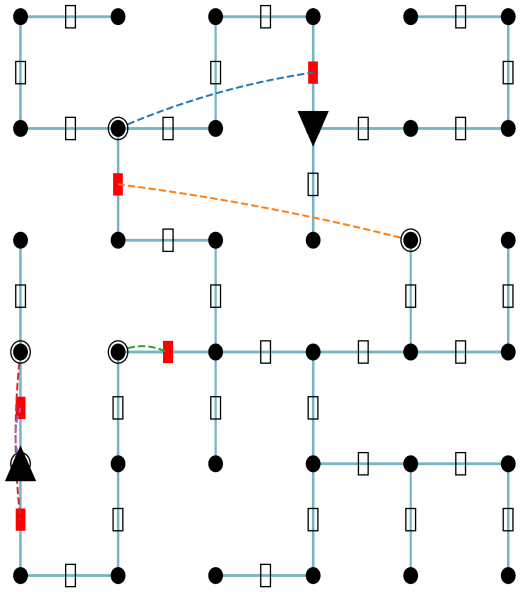

Figure 5: Example visualization of a 6×6 `seqBench` maze instance. Red rectangles denote locked doors, dashed lines indicate the locations of keys corresponding to those doors, and triangles mark the start (upward-pointing) and goal (downward-pointing) positions. This illustrates the spatial nature of the tasks.

B Prompt Design and Model Configuration Details

This appendix provides the complete details of the prompt structure and model configurations used for evaluating LLMs on the `seqBench` benchmark. The overall prompt, illustrated in Figure 6, concatenates four main components which are detailed below.

B.1 Overall Prompt Components

The prompt presented to the LLMs consists of the following components:

1. **System Instructions and Task Definition (Component 1):** Outlines the agent's task, the structure of the maze description, valid actions and their syntax, key operational constraints, and the required output format.

2. **Few-Shot Examples (Component 2):** Three examples are provided to illustrate the task, ranging in complexity. One of these examples (a simple navigation task) is detailed in Figure 6. The verbatim text for all three examples is provided in Figure 7 for completeness.

3. **Reasoning Guidance and Self-Assessment (Component 3):** Offers step-by-step algorithmic tips for solving the task and requests the model to provide a self-assessment of its confidence and the perceived difficulty of the instance.

4. **Problem Instance Facts (Component 4):** The specific natural language facts describing the current maze configuration for the task instance. As illustrated in Figure 6, these facts are appended after the preceding components and are followed by the line "YOUR SOLUTION:" to prompt the model. These facts are generated using the templates described in Appendix A.

B.2 Evaluation Metrics and Error Analysis Details

This section provides further details on specific aspects of our evaluation metrics and observed error categories, complementing the overview of metrics

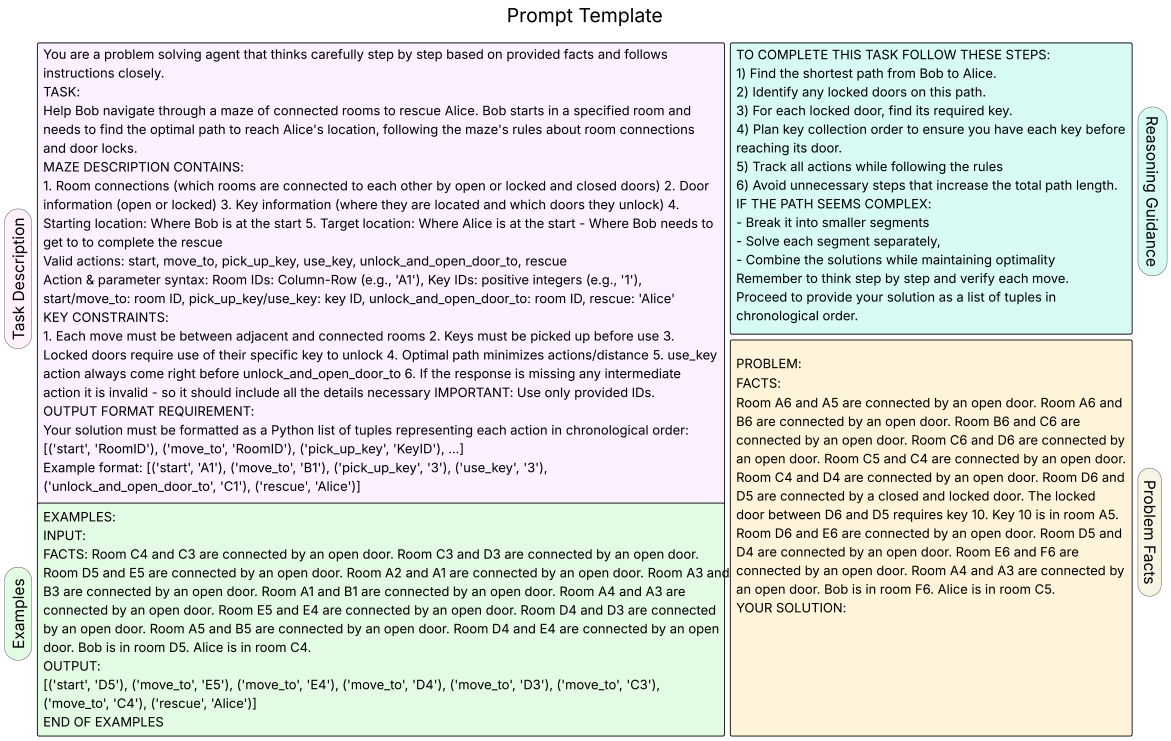

Figure 6: The complete prompt structure passed to the LLMs. This includes: Component 1 (System Instructions and Task Definition), one of the three Few-Shot Examples (Component 2, specifically a simple navigation task), Component 3 (Reasoning Guidance), and an illustration of where the Problem Instance Facts (Component 4) are inserted. For clarity and completeness, the full verbatim text for all three few-shot examples (Component 2) is provided in 7.

in Section 1 of the main paper and the discussion of failure modes in Section 2 of the main paper.

Observed Violation Categories. Failures in model solutions on seqBench tasks can be categorized into several types. Understanding these categories is crucial for interpreting model performance and failure modes. Key types of violations observed include:

- Adjacency errors (e.g., attempting to move between unconnected rooms).
- Locked door errors (e.g., navigating through locked doors without the correct key or without unlocking them).
- Key usage errors (e.g., attempting to use keys not yet collected, or using the wrong key for a door).
- Path inefficiency (e.g., taking unnecessary detours or redundant actions; while not always a hard violation that stops progress, this contributes to solutions not matching the optimal path and thus failing Pass@1).
- Missed critical actions (e.g., failing to pick up a necessary key or unlock a required door). This is a key failure mode discussed in the main paper (Section 2.4) and is often reflected in metrics like low recall or a low progress ratio if the omission occurs early and prevents further correct steps.

Identifying these distinct categories of errors provides a more granular understanding of why models fail on sequential reasoning tasks and helps in the interpretation of aggregate performance metrics reported in the main paper.

B.3 Violation Map: Qualitative Examples of Model Failures

This section provides qualitative examples of characteristic model failures to illustrate common error types. These examples visually support the discussion of failure modes in the main paper (Section 2.4, "A Key Failure Mode: Omission of Critical Steps"). Figure 8 illustrates a significant error by Gemini-2.5-Flash on a complex task, where the model generates an illegal path, bypassing neces-

1. **Example 1 (Simple Navigation):** This example, as shown in Figure 6, involves navigating a maze with only open doors.

```
EXAMPLE:
INPUT:
Maze Structure: Room C4 and C3 are connected by an open door. Room C3 and D3 are
     connected by an open door. Room D5 and E5 are connected by an open door.
    Room A2 and A1 are connected by an open door. Room A3 and B3 are connected
    by an open door. Room A1 and B1 are connected by an open door. Room A4 and
    A3 are connected by an open door. Room E5 and E4 are connected by an open
    door. Room D4 and D3 are connected by an open door. Room A5 and B5 are
    connected by an open door. Room D4 and E4 are connected by an open door. Bob
     is in room D5. Alice is in room C4.
OUTPUT:
Solution: [('start', 'D5'), ('move_to', 'E5'), ('move_to', 'E4'), ('move_to', '
    D4'), ('move_to', 'D3'), ('move_to', 'C3'), ('move_to', 'C4'), ('rescue', '
    Alice')]
```

2. **Example 2 (Single-Key Backtracking):** This example introduces a single locked door and a corresponding key.

```
EXAMPLE:
INPUT:
Maze Structure: Room A1 and A2 are connected by an open door. Room A2 and B2 are
     connected by an open door. Room B1 and B2 are connected by an open door.
    Room B1 and C1 are connected by an open door. Room C1 and C2 are connected
    by a closed and locked door. Door between C1 and C2 requires key 1. Key 1 is
     in room A2. Bob is in room A1. Alice is in room C2.
OUTPUT:
Solution: [('start', 'A1'), ('move_to', 'A2'), ('pick_up_key', '1'), ('move_to',
     'B2'), ('move_to', 'B1'), ('move_to', 'C1'), ('use_key', '1'), ('
    unlock_and_open_door_to', 'C2'), ('move_to', 'C2'), ('rescue', 'Alice')]
```

3. **Example 3 (Multi-Key Backtracking):** This example presents a more complex scenario with multiple locked doors and keys, requiring more extensive backtracking.

```
EXAMPLE:
INPUT:
Maze Structure: Room B5 and B4 are connected by a closed and locked door. The
    locked door between B5 and B4 requires key 3. Key 3 is in room B5. Room B5
    and C5 are connected by a closed and locked door. The locked door between B5
     and C5 requires key 16. Key 16 is in room C5. Room B4 and C4 are connected
    by an open door. Room C4 and C3 are connected by an open door. Room C3 and
    D3 are connected by a closed and locked door. The locked door between C3 and
     D3 requires key 10. Key 10 is in room C4. Room D5 and D4 are connected by
    an open door. Room D4 and D3 are connected by an open door. Room A5 and B5
    are connected by an open door. Bob is in room C5. Alice is in room D5.
OUTPUT:
Solution: [('start', 'C5'), ('pick_up_key', '16'), ('use_key', '16'), ('
    unlock_and_open_door_to', 'B5'), ('move_to', 'B5'), ('pick_up_key', '3'), ('
    use_key', '3'), ('unlock_and_open_door_to', 'B4'), ('move_to', 'B4'), ('
    move_to', 'C4'), ('pick_up_key', '10'), ('move_to', 'C3'), ('use_key', '10')
    , ('unlock_and_open_door_to', 'D3'), ('move_to', 'D3'), ('move_to', 'D4'),
    ('move_to', 'D5'), ('rescue', 'Alice')]
```

Figure 7: Few-shot examples provided to guide the LLMs in the maze-solving task. These examples demonstrate simple navigation, single-key backtracking, and multi-key backtracking scenarios. The three examples illustrate increasing levels of complexity.

sary steps and locked doors. This exemplifies a breakdown in multi-step planning. Additionally, Figure 9 shows another common 'adjacency error,' where a model attempts to jump between uncon-

nected rooms. This type of error reveals a critical lapse in grounding its generated actions within the spatial adjacencies explicitly stated by the task's input facts.

B.4 Quantitative Analysis of Error Patterns

To understand how and when models begin to fail within a reasoning sequence, we analyze the distribution of the first violation step. We record the time step at which the initial violation occurs in a model's generated path. Aggregating this step-indexed data across multiple instances allows us to create temporal distributions of errors. These distributions help determine whether errors tend to cluster early in the reasoning process (potentially indicating issues with initial planning or understanding of the overall problem complexity) or accumulate later (suggesting difficulties in maintaining long chains of inference or context). This analysis complements the discussion in the main paper (Section 2.4, "Path-Length Dependent First Errors: The Burden of Anticipated Complexity").

Figure 10 shows how the distribution of these first-error positions shifts with the overall problem complexity, represented by logical depth (L). As detailed in the main paper, an increase in L tends to cause errors to occur earlier in the reasoning chain.

Similarly, Figure 11 illustrates how the introduction of contextual noise (distracting facts) affects the point of failure. Increased noise also tends to precipitate earlier errors in the reasoning sequence, as discussed in the main paper in relation to sensitivity to noise (Section 2.3) and its impact on error patterns (Section 2.4).

C Supplementary Figures

This appendix provides supplementary figures that offer further visual support for analyses presented in the main paper. These figures illustrate the impact of various complexity dimensions and provide comparative views of model performance, elaborating on points made throughout Section 2 (Benchmarking Results) of the main paper.

Figure 12 details the performance of Llama-4 Maverick-17B-128E-Instruct under varying levels of noise and fact shuffling. This supports the discussion in the main paper (Section 2.3, on how these factors, especially in combination, affect success rates, with noise being a dominant factor.

To illustrate the performance consistency and disparities across different models, as detailed in Section 2.6, Figure 13 presents scatter and density plots of mean progress ratios. These plots clearly demonstrate that model performance hierarchies are not strictly linear. They reveal 'performance inversions'—instances, also noted in Section 2.6, where models with typically lower overall performance (e.g., lower average L_0) occasionally solve specific complex problems that models with higher average L_0 values fail on.

Figure 14 isolates the impact of shuffle ratio on model performance when other factors like noise are controlled. This visualization corresponds to the findings discussed in the main paper (Section 2.3, "Fact Ordering (Shuffle Ratio)") that simple reordering of facts has a minimal impact on the performance of the evaluated models under low-noise conditions.

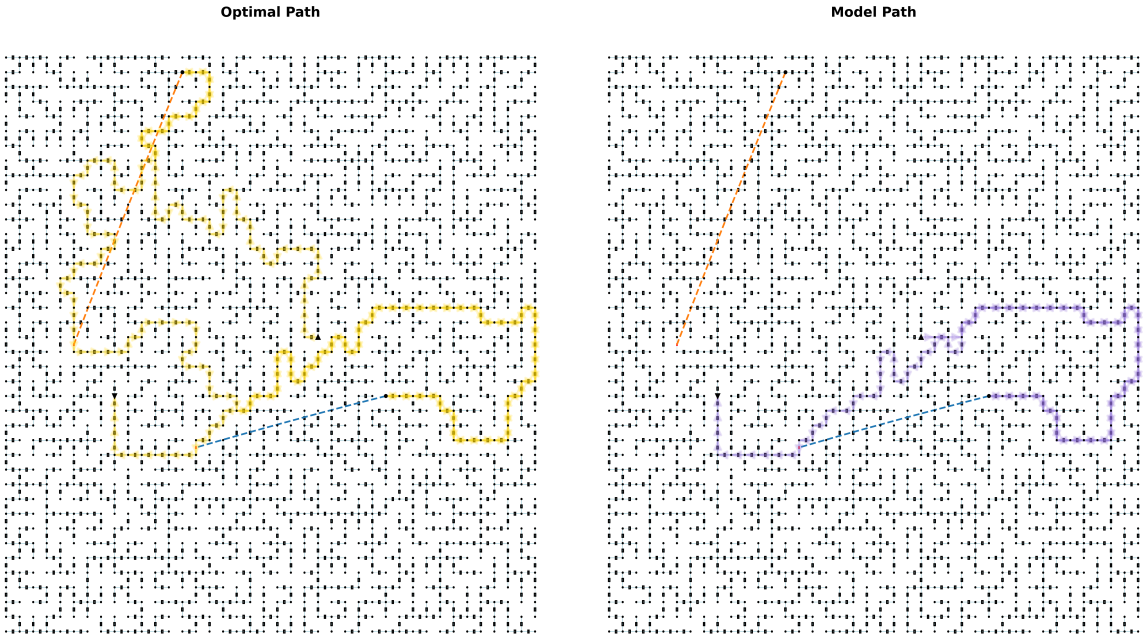

Figure 8: Illustrative failure case for Gemini-2.5-Flash on a 40x40 task. Left: Optimal path (yellow). Right: Model's generated path showing an illegal adjacency jump (red arrow), bypassing multiple rooms and a locked door, despite only supporting facts being provided. This highlights a breakdown in multi-step planning.

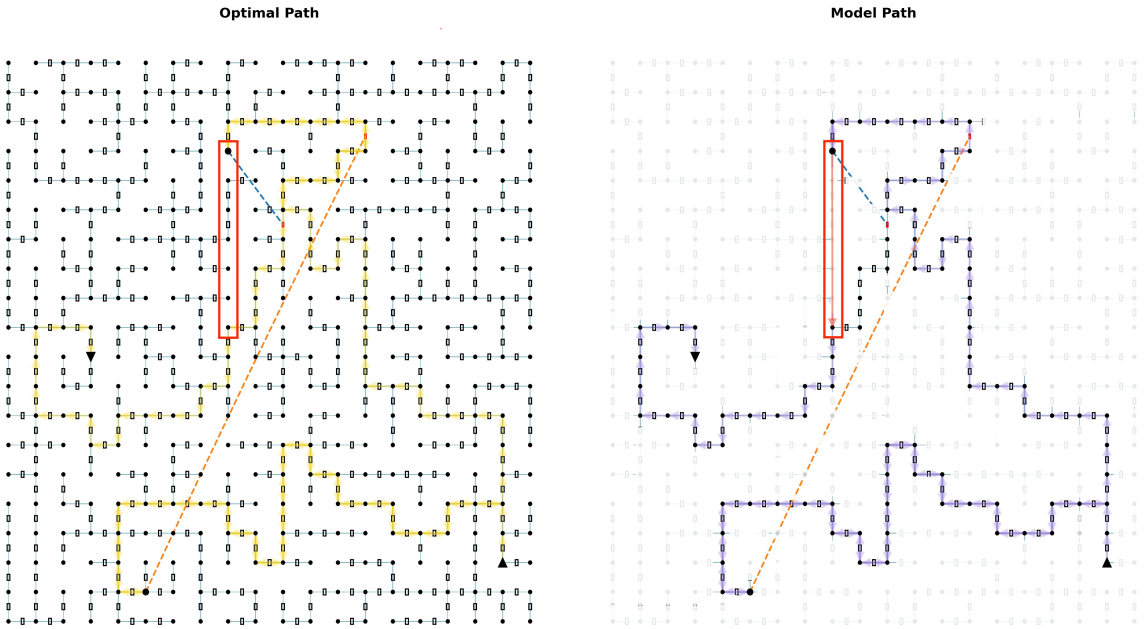

Figure 9: Illustrative failure case of an 'adjacency error' in model-generated pathfinding. The left panel displays the optimal path (yellow) to the target (triangle). The right panel shows a suboptimal path (purple) generated by the model. This example highlights a common error where, after a sequence of actions (in this scenario, following a key acquisition), the model fails to navigate through valid connections. Instead, it attempts to 'jump' directly between two unconnected rooms. This violation of room adjacency constraints is a key challenge in model performance.

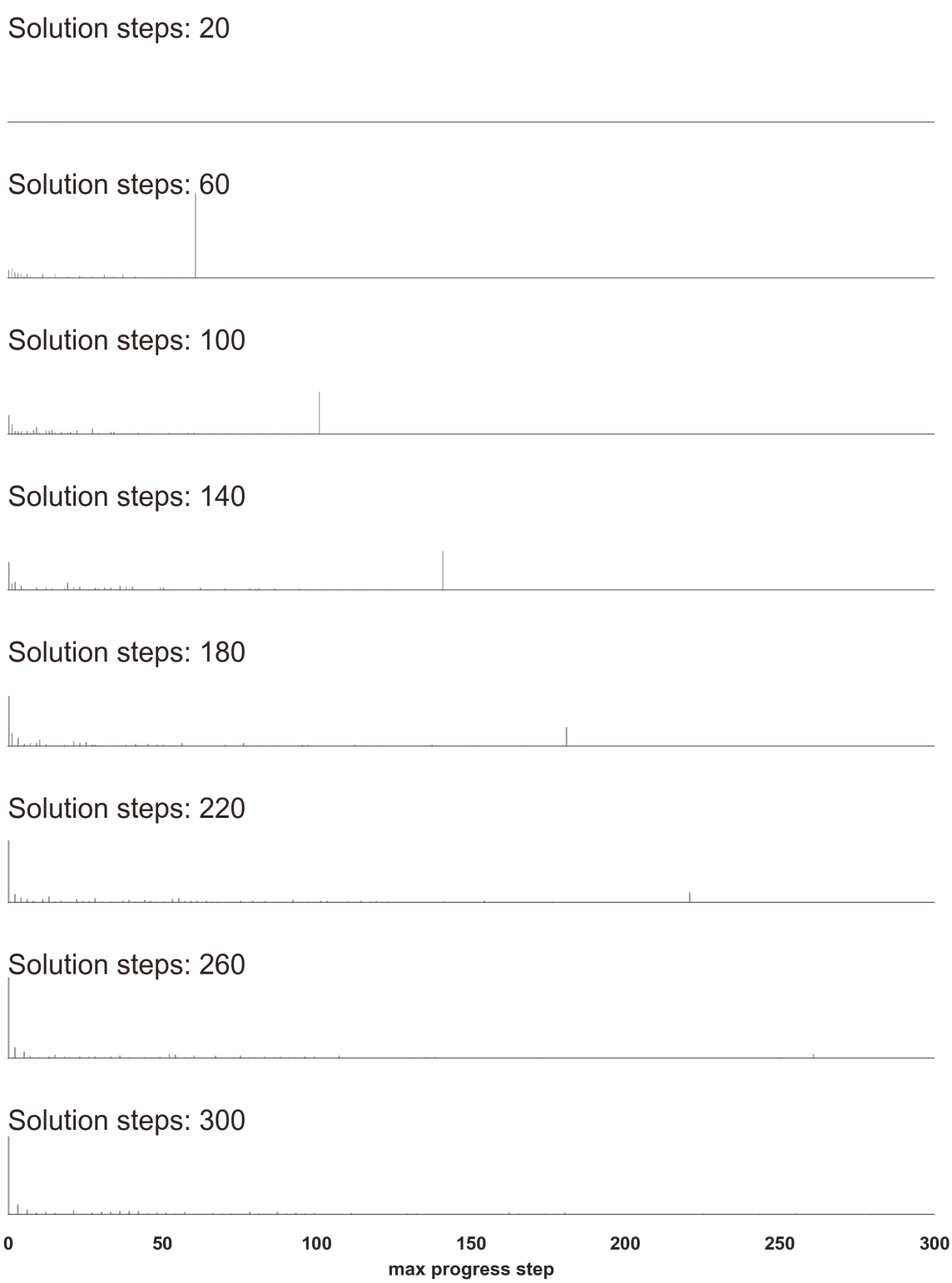

Figure 10: Distribution of first-violation steps for Gemini-2.5-Flash across varying logical depths (L). As L (total required path length) increases, the distribution of first errors tends to shift leftward, indicating that models are more likely to fail at earlier steps in longer problems. This suggests that anticipated global complexity impacts reasoning from the outset.

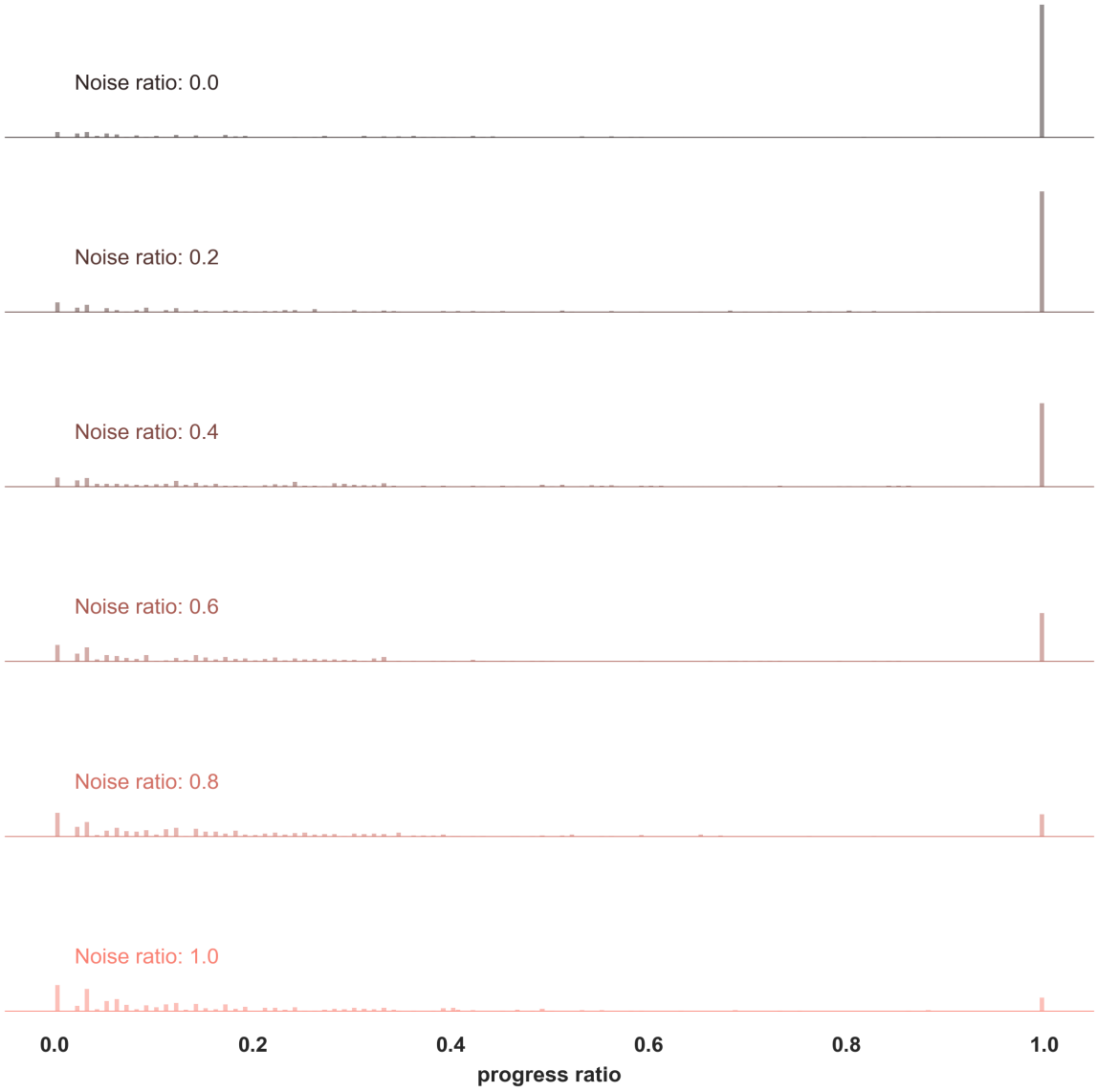

Figure 11: Impact of increasing noise ratio on the distribution of failure steps for Gemini 2.5 Flash. As noise (proportion of distracting facts) increases, failures tend to occur earlier in the reasoning chain. This reflects increased difficulty in isolating relevant information and maintaining focus.

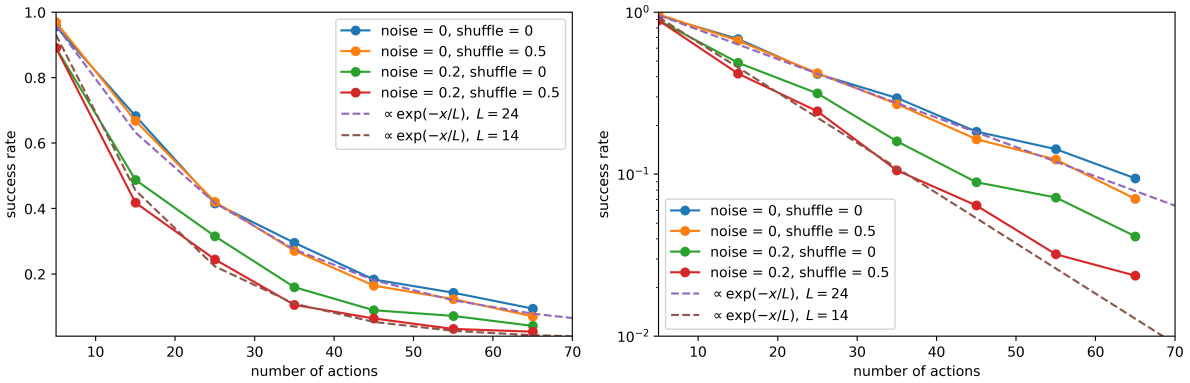

Figure 12: Pass@1 success rate for Llama-4 Maverick-17B-128E-Instruct versus solution length (L) under different noise and shuffle ratios. Left: Linear scale. Right: Log-linear scale. Performance degrades with increased noise but is less affected by shuffle ratios.

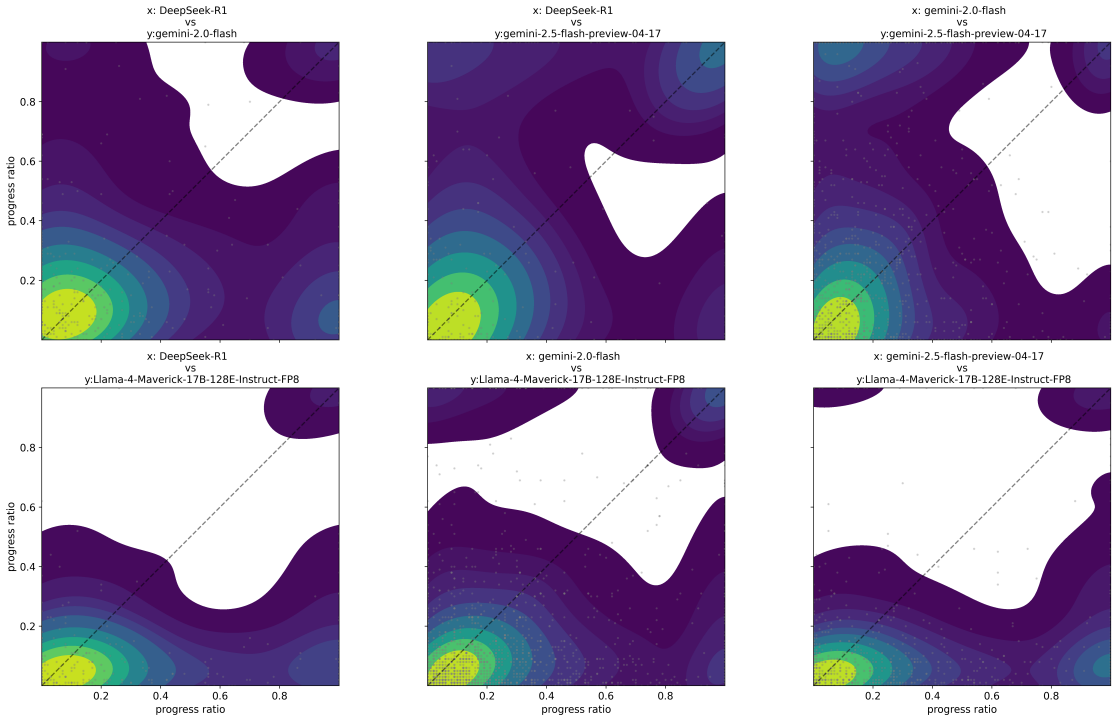

Figure 13: Scatter and density plots of progress ratios per task instance, comparing model pairs on the tasks. These plots illustrate performance agreement and disparities on the same instances of pathfinding tasks. Notably, Gemini-2.5-Flash (example) often succeeds on instances where other models achieve near-zero progress. Data from experiments in Figure 1 (main paper).

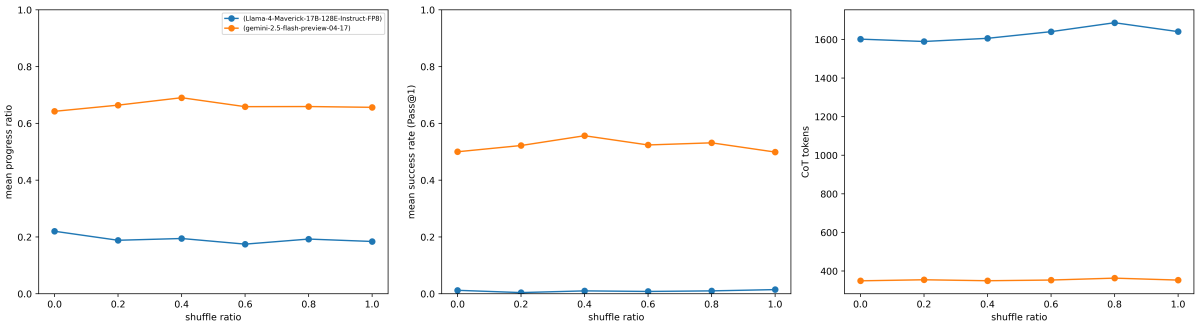

Figure 14: Impact of shuffle ratio on Pass@1 success rate. Varying the degree of mixing (shuffle) between supporting and distracting facts shows minimal impact on performance for Gemini 2.5 Flash and Llama-4 Maverick, suggesting robustness to fact order when noise is controlled. The generation and sampling of maze instances for these tasks follow the same methodology detailed for experiments in the main paper (Figures 3 and 4).