MedVerse: Efficient and Reliable Medical Reasoning via DAG-Structured Parallel Execution

Jianwen Chen[1*], **Xinyu Yang**[2*], **Peng Xia**[1] **Arian Azarang**[1] **Yueh Z Lee**[1]
Gang Li[1] **Hongtu Zhu**[1] **Yun Li**[1] **Beidi Chen**[2] **Huaxiu Yao**[1]
[1]UNC-Chapel Hill [2]Carnegie Mellon University
[*]Equal contribution

Abstract

Large language models (LLMs) have demonstrated strong performance and rapid progress in a wide range of medical reasoning tasks. However, their sequential autoregressive decoding forces inherently parallel clinical reasoning, such as differential diagnosis, into a single linear reasoning path, limiting both efficiency and reliability for complex medical problems. To address this, we propose MedVerse, a reasoning framework for complex medical inference that reformulates medical reasoning as a parallelizable directed acyclic graph (DAG) process based on Petri Net theory. The framework adopts a full-stack design across data, model architecture, and system execution. For data creation, we introduce the MedVerse Curator, an automated pipeline that synthesizes knowledge-grounded medical reasoning path and transforms them into Petri Net–structured representations. At the architectural level, we propose a topology-aware attention mechanism with adaptive position indices that supports parallel reasoning while preserving logical consistency. Systematically, we develop a customized inference engine that supports parallel execution without additional overhead. Empirical evaluations show that MedVerse improves strong general-purpose LLMs by up to 8.9%. Compared to specialized medical LLMs, MedVerse achieves comparable performance with improved clinical reliability, while delivering a 1.3× reduction in inference latency and a 1.7× increase in generation throughput, enabled by its parallel decoding capability. Code is available at https://github.com/aiming-lab/MedVerse.

1 Introduction

Recent advancements in Large Reasoning Models (LRMs) (OpenAI, 2024; Guo et al., 2025) have broadened the capabilities of medical artificial intelligence (Moor et al., 2023a; Thirunavukarasu et al., 2023; Nori et al., 2023; Xia et al., 2024a), enabling a transition beyond simple information retrieval toward complex clinical reasoning (Xia et al., 2024b, 2025b; Alam et al., 2025; Zhang et al., 2025). In particular, state-of-the-art medical LRMs, such as MedReason (Wu et al., 2025), HuatuoGPT-o1 (Chen et al., 2024), and m1 (Huang et al., 2025a), have shown that Chain-of-Thought (CoT) (Wei et al., 2022) reasoning can significantly enhance diagnostic accuracy. However, it remains unknown whether such models can be reliably and efficiently deployed in real-world clinical settings.

In practice, physicians increasingly employ these models to assist with clinical decision-making. Yet, the sequential nature of autoregressive (AR) models is misaligned with human cognitive processes, which naturally considers multiple differential diagnoses simultaneously (Kassirer and Gorry, 1978). Accordingly, this mismatch leads to three fundamental limitations: (i) *Accuracy:* linear reasoning narrows the diagnostic hypothesis space, limiting the exploration of alternative diagnostic pathways; (ii) *Efficiency:* serialized decoding increases response latency, posing challenges for real-time clinical decision support; (iii) *Interpretability:* unstructured CoT lacks explicit causal relationships, hindering clinical interpretation and validation. An overview of these challenges is shown in Figure 1.

To address these bottlenecks, prior work on general-purpose LRMs has proposed parallel thinking (Luong et al., 2025; Wang et al., 2025; Ding et al., 2025), enabling multiple reasoning branches to be explored concurrently before being synthesized into a final conclusion. However, most existing parallel generation frameworks induce parallelism primarily through brute-force repeat sampling at the early stages of generation (Brown et al., 2024). Such approaches fail to capture the complex structure in clinical inference that involves multiple competing hypotheses and conditional dependencies (Elstein et al., 1978), resulting in suboptimal token efficiency and limited structural interpretabil-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 15320–15336

ity. Moreover, these inference-only solutions cannot internalize parallel thinking into the model, due to the absence of data-centric, end-to-end learning. While recent studies (Yang et al., 2025; Pan et al., 2025) enables recursive and consecutive parallel thinking patterns, they are limited to serial–parallel graph structures, failing to adapt either structure or knowledge required for complex clinical reasoning.

In this work, we overcome these challenges with **MedVerse**, a novel modeling framework that enables native directed acyclic graph (DAG)-based parallel generation tailored for medical reasoning. Specifically, we formalize the entire differential diagnosis process with Petri Net (Petri, 1962), a bipartite graph composed of places and transitions. In this formulation, each place corresponds to a clinical entity (e.g., symptoms), while each transition encodes a directed relation between entities.

Built upon this structure, MedVerse is realized through a co-design of data, algorithm, and system:

- **Data Curation.** We propose *MedVerse Curator*, an automated LLM-assisted pipeline that synthesizes knowledge-grounded medical reasoning trajectories. Starting from an entity-level knowledge graph extracted for each query, our curator leverages the Petri Net representation to transform it into a transition-level DAG, in which each node refers to an atomic reasoning step between entities. In practice, this process yields **MedVerse-14K**, a training corpus of 13,904 high-quality structured medical reasoning examples.

- **Algorithm Design.** We propose *MedVerse Attention*, a new attention mechanism for DAG-based execution. This is achieved by modifying attention masks and position embeddings to strictly ensure the DAG-structured dependency. This design also excels in data efficiency: since these changes are minor, pre-trained AR models can be rapidly fine-tuned from causal attention to MedVerse attention through a few thousand examples.

- **System Implementation.** We propose *MedVerse Engine*, a high-performance serving engine built upon the Multiverse Engine (Yang et al., 2025). Our inference starts from a linear planning stage that generates the DAG-structured plan. Next, our engine dynamically extract this plan to enable true parallel decoding while respecting the underlying topological dependencies. Our engine enables parallel generation with negligible cost via continuous batching and radix attention.

Figure 1: Limitations of sequential chain-of-thought reasoning in medical diagnosis. (a) **Accuracy:** Linear execution suffers from contextual pollution due to early incorrect hypotheses (red), whereas parallel reasoning preserves correct inference paths (green). (b) **Efficiency:** Sequential reasoning repeatedly processes overlapping evidence, leading to redundant computation (orange). (c) **Interpretability:** Unstructured chain-of-thought (red) obscures explicit causal dependencies due to a structural mismatch with parallel clinical reasoning.

Empirical evaluations on clinical benchmarks show that MedVerse improves accuracy by 4.8% on Qwen2.5-7B and 8.9% on Llama-3.1-8B, matching the performance of specialized reasoning models such as MedReason and HuatuoGPT-o1. Beyond accuracy, MedVerse overcomes the serial constraints of autoregression, achieving a 1.3× speedup in inference latency and a 69.3% increase in peak throughput.

2 Related Work

LLMs in Medical Reasoning. The application of LLMs in healthcare has evolved from general domain adaptation to specialized clinical reasoning. Early efforts, such as Med-PaLM (Singhal et al., 2023) and PMC-LLaMA (Wu et al., 2023), focused on injecting medical knowledge via continual pre-training on biomedical corpora. Subsequently, instruction-tuned models like HuatuoGPT (Zhang

et al., 2023) and ChatDoctor (Li et al., 2023b) leveraged real-world dialogue data to align models with physician behaviors (Li et al., 2023a; Moor et al., 2023b; Nath et al., 2025). More recently, the focus has shifted towards enhancing diagnostic logic, e.g., models trained on MedReason are explicitly supervised to generate CoT rationales (Wu et al., 2025; Xia et al., 2025a; Zhu et al., 2025; Lai et al., 2025; Huang et al., 2025b). Despite these advances, existing models remain fundamentally constrained by the autoregressive decoding mechanism (Vaswani et al., 2017). They generate rationales as a rigid, serial token sequence, yielding an inference complexity of $\mathcal{O}(N)$. This computational linearity not only imposes high latency in real-time scenarios but also restricts the model to a single narrative thread, making it inefficient to maintain context across long, complex reasoning trajectories.

Parallel Generation and Efficient Inference To break the serial decoding barrier, architectural approaches like Non-Autoregressive (NAR) decoding (Gu et al., 2017) and Speculative Decoding (Leviathan et al., 2023) attempt simultaneous token generation, though often facing coherence or verification bottlenecks. Notably, Multiverse introduces a MapReduce paradigm, decomposing generation into parallel "Map" branches merged via "Reduce" steps, supported by a specialized engine leveraging Radix Attention (Zheng et al., 2024) for efficiency. However, these general-purpose paradigms lack medical grounding, and their topologies are constrained to simple fork-join patterns. This structural simplicity fails to capture the complex, non-linear branching networks of differential diagnosis (Bowen, 2006). MedVerse addresses this limitation by co-designing a medically-grounded architecture with a custom vLLM-based engine (Kwon et al., 2023), utilizing Radix Attention to enable zero-copy forking specifically tailored for complex medical reasoning topologies.

3 MedVerse Modeling

This section introduces MedVerse, a novel modeling framework that reimagines the entire clinical reasoning process as DAG-based execution rather than sequential generation via a Petri Net abstraction, thereby improving reliability and efficiency.

3.1 DAG Structure for Medical Reasoning

As illustrated in Figure 2, multiple diagnostic hypotheses may share intermediate evidence or converge on common pathological mechanisms, leading to complex DAG-structured dependencies across different entities for medical reasoning. Such structures extend beyond the capabilities of sequential CoT reasoning or the tree-based extensions (e.g., Tree-of-Thoughts (Yao et al., 2023)). Motivated by this observation, we formalize medical reasoning using a DAG $\mathcal{G} = (V, E)$ as follows:

- **Nodes as Reasoning States (V):** Each node represents an intermediate reasoning state. Specifically, we distinguish three types of nodes: (i) *source nodes*, corresponding to clinical entities grounded in the input question, which only have outgoing edges; (ii) *hypothesis nodes*, representing diagnostic hypotheses or pathological states that may both *split* into multiple downstream paths and *merge* evidence from multiple upstream paths; and (3) *conclusion nodes*, referring to final diagnostic outcomes, which only admit incoming edges and serve as unique convergence points of the entire reasoning process.
- **Edges as Reasoning Steps (E):** Each (directed) edges encode conditional dependencies between reasoning states. An edge (u, v) indicates an admissible reasoning step from state u to state v. To ensure structural acyclicity, all edges are restricted to forward dependencies that follow the temporal and causal order of clinical reasoning.

Figure 2 showcases that our formulation naturally captures DAG structures in clinical reasoning.

3.2 Extension to Petri Nets

While the DAG structure provides a topological blueprint, its static representation of solitary states fails to capture the generative transition logic essential for contextualizing history-dependent LLM inference. To realize this structure into an executable parallel process, we formalize our framework using Petri Nets. This mathematical grounding bridges the gap between the logical graph $\mathcal{G}$ and the physical execution of LLM inference, enabling reasoning states to be instantiated as places and causal dependencies as transitions for parallel execution (Figure 2).

Formal Definition. We define the execution model as a tuple $\mathcal{N} = (P, T, F, M_0)$, explicitly mapping to the DAG components:

- $P = \{p_1, \ldots, p_m\}$ is a finite set of *places*, corresponding to the nodes (V) in the DAG. Functionally, a place serves as a state container, holding the context or belief state waiting to be processed.

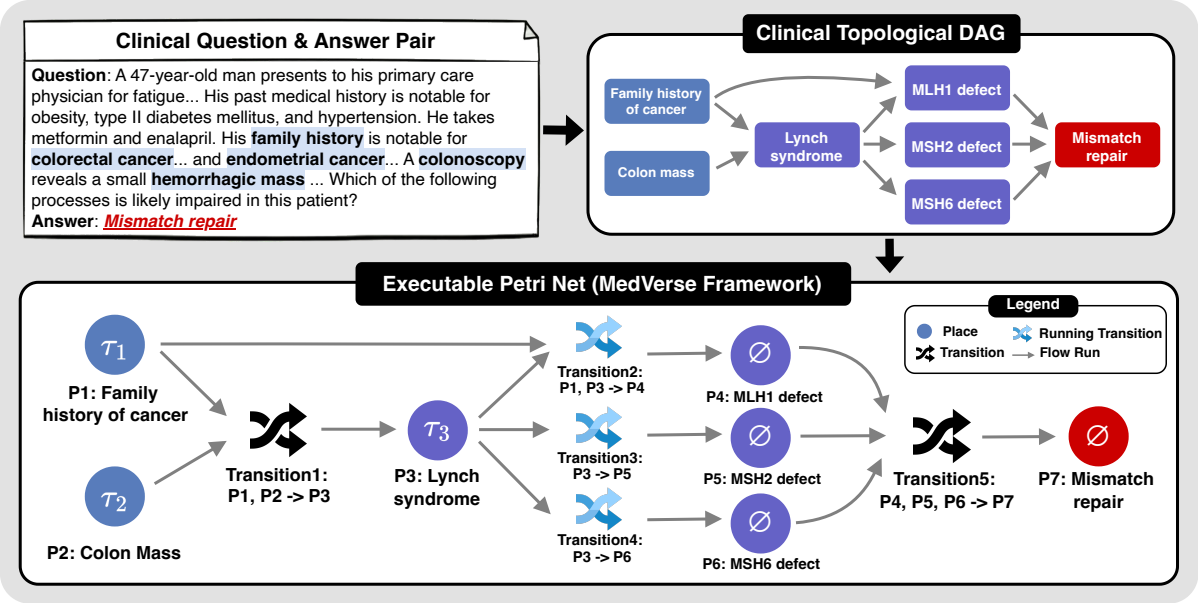

Figure 2: Illustration of the topological modeling process. The framework first extracts a structured clinical topological DAG from an unstructured question-answer pair, capturing causal dependencies. This DAG is then formally mapped into an Executable Petri Net, where reasoning states are instantiated as places and their dependency relations are realized through transitions.

- $T = \{t_1, \ldots, t_n\}$ is a finite set of *transitions*, representing reasoning steps. Edges are mapped via many-to-one aggregation: converging edges (e.g., $A, B \to C$) form a single transition, while divergent edges ($A \to B, A \to C$) instantiate distinct transitions.
- $F \subseteq (P \times T) \cup (T \times P)$ defines the flow arcs that adhere to the DAG's topological direction. We denote the set of input places for a transition t as the pre-set $\bullet t = \{p \mid (p, t) \in F\}$ and the output places as the post-set $t\bullet = \{q \mid (t, q) \in F\}$.
- M_0 is the initial marking such that for any place $p \in P$ corresponding to a DAG node with in-degree zero, $M_0(p)$ is non-empty, while $M_0(p) = \emptyset$ for all other places.

MedVerse Token Semantics. Standard Petri nets treat tokens as simple counters. To support the rich context of medical reasoning, we instantiate $\mathcal{N}$ as a Colored Petri Net (CPN). We define a token not as a scalar, but as a semantic tuple $\tau = (\mathbf{h}, \mathbf{k})$:

- $\mathbf{h}$: Encapsulates the textual history generated along the current path.
- $\mathbf{k}$: Denotes the KV-cache indices associated with that history.

This definition transforms tokens into computational state carriers, enabling the system to pass memory references rather than copying full text, which is pivotal for efficiency.

Execution as Inference. Inference is modeled as token flow through the Petri Net. A transition t is enabled when tokens are present in all its input places and its output places are empty, ensuring each reasoning step executes exactly once. When fired, t invokes the LLM to generate the corresponding reasoning output, reading from the input tokens and producing new tokens at its output places. Each resulting token inherits and extends both the textual history $\mathbf{h}$ and KV-cache references $\mathbf{k}$: the engine appends the newly generated text to $\mathbf{h}$ and maps the corresponding memory blocks to $\mathbf{k}$. Multiple enabled transitions may fire concurrently, yielding parallel decoding streams.

3.3 Execution Semantics

Given the executable Petri Net constructed in Section 3.2, we specifies the execution semantics of MedVerse. We decouple *scheduling* from *execution*: scheduling identifies the set of causally enabled transitions ready for execution, while execution operates on this set via two abstract primitives, **Fork** and **Join**, to realize parallel generation with explicit causal synchronization.

Scheduling via Enabled-Transition Frontier. At runtime, execution is governed by the current Petri Net marking M_k, which encodes token availability at each place. A transition is schedulable if tokens are present in all its input places and all output places are empty. The enabled-transition frontier

at step k is defined as

$$\mathcal{F}_k = \Big\{ t \in T \;\Big|\; \big(\forall p \in {\bullet t}, M_k(p) \neq \emptyset\big) \wedge \big(\forall q \in t{\bullet}, M_k(q) = \emptyset\big) \Big\}, \tag{1}$$

which represents the maximal set of transitions that can be executed concurrently without violating causal dependencies.

Fork and Join Primitives. Given the frontier $\mathcal{F}_k$, the engine executes transitions based on their topological dependencies. **Fork** is applied to transitions sharing a common predecessor, initiating parallel decoding streams that inherit the exact same context. Conversely, **Join** is applied to transitions aggregating multiple reasoning branches, performing logical synchronization by merging distinct predecessor histories before generation proceeds.

Execution Loop. After all transitions in $\mathcal{F}_k$ complete, the marking advances to M_{k+1}, the frontier is recomputed, and the scheduling–execution cycle repeats until no further transitions are enabled.

3.4 Structured Generation Flow

Building on the structured representations and execution semantics introduced in Sections 3.1–3.3, MedVerse organizes LLM generation into a structured three-stage flow: planning, execution, and conclusion (see Figure 3). The process first constructs an explicit reasoning plan, executes it under graph-based constraints, and finally aggregates the results into a unified conclusion.

Planning Stage. Generation begins with a planning stage that reconciles sequential generation with non-linear reasoning via a *Think–then–Map* strategy. The model first generates multiple linear reasoning paths connecting clinical evidence to candidate diagnoses. These paths collectively encode overlapping dependencies that implicitly form a logical DAG. The model then consolidates them into a structured `<Plan>` block, where reasoning steps are organized using `<Outline>` tags annotated with dependency information. These annotations instantiate the flow relation F of the Petri Net, specifying the reasoning topology while deferring detailed inference to execution.

Execution Stage. Given the specified plan, generation proceeds within the `<Execution>` block following the execution paradigm in Section 3.3. At each iteration, the enabled-transition frontier is identified from the current marking. The engine then concurrently executes the transitions within this frontier by applying **Fork** and **Join**, encapsulating the generated reasoning for each transition within `<Step>` tags.

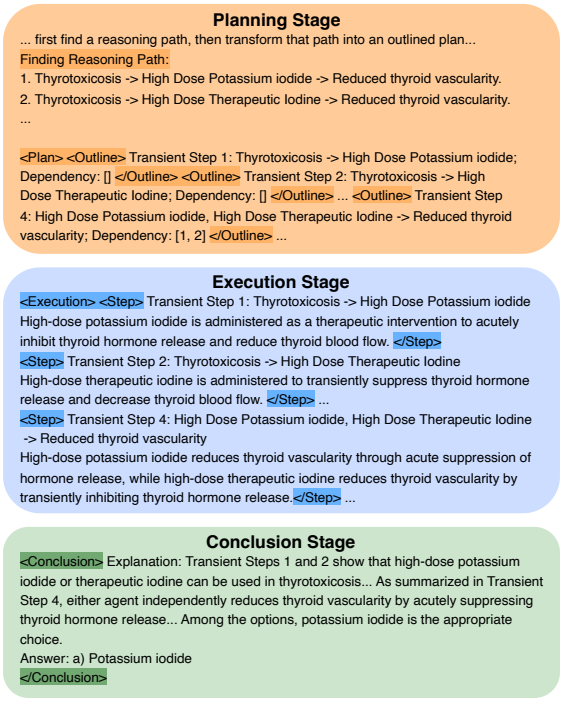

Figure 3: Example of the structured generation flow in MedVerse. The reasoning process is explicitly decomposed into planning, execution, and conclusion stages.

Conclusion Stage. After all executable reasoning steps have completed, the process transitions to a final synthesis phase marked by the `<Conclusion>` block. In this phase, the outcomes of all completed reasoning branches are aggregated to produce a unified diagnostic conclusion.

4 MedVerse Instantiation

To instantiate our MedVerse model discussed in Section 3, we present a comprehensive suite including three core components: the **MedVerse Curator** for structured data generation, the **MedVerse Attention** for DAG-structured modeling, and the **MedVerse Engine** to enable parallel execution.

4.1 Data Curation: MedVerse Curator

We introduce the MedVerse Curator, an LLM-assisted pipeline that extracts knowledge-grounded reasoning paths from a medical knowledge graph and compiles them into structured training instances executable under the MedVerse framework. Specifically, it includes four steps.

I. Knowledge-Grounded Retrieval. The curator grounds medical reasoning in established clinical knowledge by first mapping questions and answer

candidates to standardized medical concepts. It then retrieves plausible reasoning paths connecting these concepts through a medical knowledge graph, and prunes irrelevant or low-confidence branches.

II. Topological Planning. The resulting linear reasoning skeletons are refined into executable plans using a specialized prompt that removes redundancy and enforces logical correctness and coherence. A DAG validity check ensures that all dependencies are acyclic; paths with invalid structures are discarded or re-routed.

III. Structural Synthesis. Given a topology, the curator generates structured data in the MedVerse format. An LLM produces step-by-step reasoning for each execution transition, followed by an automated refinement module that smooths transitions across branches and ensures reasoning correctness. Finally, a coherent conclusion is synthesized from the refined execution trajectory.

IV. Dual-Layer Verification. Finally, syntax-level validation and model-based logical evaluation are applied. Samples failing either check are iteratively regenerated until all constraints are satisfied.

MedVerse-14K Dataset. In practice, we apply our automated pipeline to the training subset of multiple medical training datasets, including MedQA (Jin et al., 2021), MedMCQA (Pal et al., 2022), PubMedQA (Jin et al., 2019), MMLU-Medical (Hendrycks et al., 2021a,b), HuatuoGPT-o1, MedXpert (Zuo et al., 2025), and Humanity's Last Exam (HLE) (Phan et al., 2025). This process yields **MedVerse-14K** that comprises 13,904 high-quality and structured reasoning trajectories, covering complex long-context medical problems.

4.2 Algorithm Design: MedVerse Attention

Next, we introduce **MedVerse Attention** to replace standard causal attention (Vaswani et al., 2017). The causal attention computes the i-th token's output with query $\boldsymbol{q}_i$, and keys $\boldsymbol{k}_j$, values $\boldsymbol{v}_j$ ($j \le i$):

$$a_{ij} = \text{Softmax}\Big((\boldsymbol{q}_i \cdot P(i))^\top (\boldsymbol{k}_j \cdot P(j)) + M_{ij}\Big). \tag{2}$$

where $M_{ij} = \begin{cases} 0, & j \le i \\ -\infty, & \text{otherwise} \end{cases}$ is casual attention mask and $P(i)$ is embedding for position i.

DAG-based Attention Mask. Transitions within the same enabled-transition frontier may execute concurrently and must remain causally independent. To enforce this property, we construct a layer-wise mutual exclusion mask M. During training, the input sequence is segmented into frontier layers according to the execution plan, where each layer contains a set of parallel reasoning steps $S_1, S_2, \ldots$. For two tokens i and j associated with steps S_u and S_v, the attention bias is defined as:

$$M_{ij} = \begin{cases} -\infty & \text{if } j > i \\ -\infty & \text{if } \text{Layer}(i) = \text{Layer}(j) \wedge S_u \neq S_v \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The first condition preserves standard autoregressive causality, while the second prevents information leakage between parallel transitions.

Adaptive Position Indices. While masking enforces causal isolation, standard monotonic position indices cannot represent Petri Net execution synchronization. We therefore assign position indices based on the logical execution timeline rather than linear token order. Tokens generated by transitions within the same enabled-transition frontier are assigned an identical starting index (fork alignment). For transitions that join multiple branches in a subsequent frontier, the position index is set to the maximum index among all predecessor branches, allowing the model to attend to the complete causal history.

Together, topology-aware masking and adaptive position indices implement the execution semantics of Section 3.3 within standard autoregressive Transformers, without modifying the training pipeline.

4.3 Inference Engine: MedVerse Engine

To translate the execution semantics of the Petri Net into practical inference speedups, we develop the MedVerse Engine, a high-performance serving system built upon the Multiverse Engine. Unlike standard engines that treat generation as a uniform stream, our engine implements a Hybrid Execution Pipeline that seamlessly transitions from linear LLM-driven planning to system-guided parallel execution. This engine includes two phases, which are detailed as follows:

Phase I: Linear Planning & Graph Initialization. The engine first employs standard autoregressive decoding to perform linear planning, during which the LLM translates the input into multiple linear reasoning paths followed by a topological `<Plan>` block. This phase is managed strictly as a conventional linear generation process to ensure logical correctness and coherence. Upon detecting the `</Plan>` tag, the engine pauses generation and

Table 1: Performance comparison on medical reasoning benchmarks. We report accuracy (%) across different datasets. **Bold** indicates the best performance.

	Qwen2.5-7B-Instruct			LLaMA-3.1-8B-Instruct			
Benchmark	Original	MedReason	**MedVerse**	Original	MedReason	Huatuo-o1	**MedVerse**
HLE (Biomed)	18.4	**20.8**	19.6	13.6	20.2	14.6	**20.6**
MedBullets (op4)	45.8	49.7	**55.2**	48.7	57.1	55.8	**62.3**
MedBullets (op5)	39.6	44.2	**48.0**	42.5	51.0	**53.9**	53.6
MedQA	56.2	56.2	**58.6**	58.7	63.9	**72.4**	66.4
MedXpert	12.3	14.5	**15.3**	13.2	18.4	16.8	**19.3**
Average	34.5	37.1	**39.3**	35.3	42.2	42.7	**44.2**

parses the dependency annotations specified in the <Outline> tags to instantiate the in-memory Petri Net structure $\mathcal{N}$ and initialize the token marking M_0, thereby triggering an immediate transition to graph-based execution mode.

Phase II: Frontier-Based Graph Execution. Guided by the parsed topology, the scheduler identifies the enabled-transition frontier $\mathcal{F}_k$ (Sec. 3.3) at each step and executes all transitions in the frontier concurrently using two memory-optimized primitives. For transitions that branch from a common predecessor, the engine applies *Fork* execution, spawning multiple parallel decoding streams that share the same prefix KV cache via Radix Attention, thereby enabling zero-copy prefix reuse until each corresponding <Step> block completes. Conversely, for transitions with multiple predecessors, execution is deferred until all upstream reasoning paths finish, after which the engine applies *Join* execution by merging the KV states of all predecessor paths together with the preceding context to construct a unified KV cache. Leveraging the flexible radix cache layout, this merge is performed without padding or physical memory copying, and generation then proceeds from the merged KV state along the newly constructed sequence.

5 Experiments

5.1 Setup

Training. We fine-tune MedVerse variants from the Qwen2.5-7B-Instruct (Team et al., 2024) and Llama-3.1-8B-Instruct (Dubey et al., 2024) checkpoints, incorporating our proposed MedVerse attention mechanism. Training is conducted on the curated MedVerse-14K dataset. All base models are trained for 3 epochs with a learning rate of 10^{-5} and a batch size of 128. Fine-tuning is conducted using 4 NVIDIA H200 GPUs with PyTorch FSDP.

Evaluation. We evaluate all MedVerse variants across five standard medical reasoning benchmarks, including MedQA, MedXpert, Humanity's Last Exam (HLE) (Phan et al., 2025), and MedBullets (op4, op5) (Chen et al., 2025). All experiments are conducted using our SGLang-based MedVerse Engine.

Baselines. We compare MedVerse against the original base models and other medical LLMs, including MedReason-8B and HuatuoGPT-o1-RL-8B. In addition to accuracy, we report both latency and throughput to assess efficiency gains of MedVerse. For fair comparisons, MedReason-8B is fine-tuned using the same number of examples as MedVerse-14K, while following its official training recipe.

5.2 Performance Evaluation

Table 1 reports the performance of MedVerse on medical reasoning benchmarks. Across both backbones, MedVerse consistently outperforms the base models and strong autoregressive medical LLM baselines. For Qwen2.5-7B-Instruct, MedVerse improves the average accuracy from 34.5% to 39.3%, exceeding MedReason (37.1%). For LLaMA-3.1-8B-Instruct, MedVerse achieves the highest average accuracy of 44.2%, surpassing both MedReason (42.2%) and HuatuoGPT-o1-RL-8B (42.7%). MedVerse further scales to 14B with a +18.0% relative gain and consistent 1.32× latency speedup (Appendix A.2).

5.3 Efficiency Analysis

To validate the efficiency of our system, we conducted a rigorous efficiency evaluation comparing MedVerse against standard autoregressive baselines on NVIDIA H200 GPUs. Our analysis focuses on three key dimensions: the latency for generating full reasoning chains, the engine's throughput under controlled output lengths, and the compu-

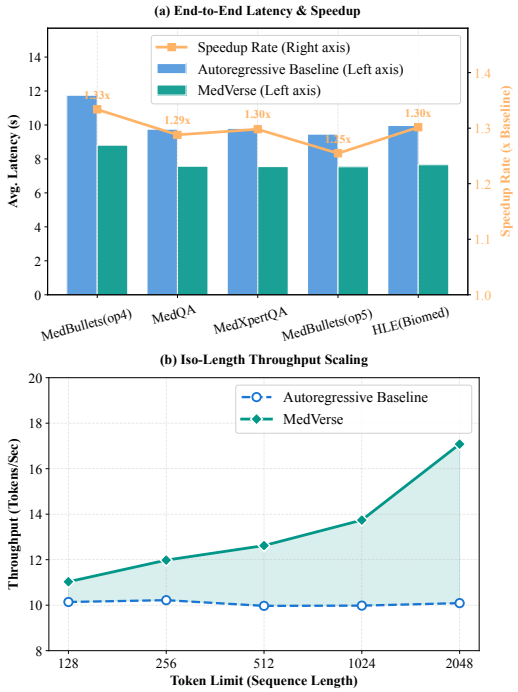

Figure 4: **Efficiency Metrics.** (a) Average latency and relative speedup (orange line) across five datasets. MedVerse consistently outperforms the baseline. (b) Throughput vs. Sequence Length. Our method exhibits better scaling properties, maintaining higher throughput as token complexity increases.

tational cost decomposition across pipeline stages.

End-to-End CoT Latency. First, we measured the total wall-clock time required to answer medical queries across five diverse datasets (e.g., MedXpertQA, MedQA) with a batch size of 64. As shown in Figure 4(a), MedVerse consistently outperforms the sequential AR baseline, achieving a stable speedup ranging from 1.25× to 1.33× (indicated by the orange trend line). Theoretically, in standard AR models, generating a comprehensive diagnosis with multiple branches requires linear time $\mathcal{O}(N)$, where N is the total token count. In contrast, MedVerse leverages its topological structure to generate independent reasoning paths simultaneously. This shifts the latency complexity from total length to the *topological depth* of the reasoning graph $\mathcal{O}(D)$, drastically reducing user wait time for complex, multi-step queries.

Iso-Length Throughput Comparison. To isolate the system's raw generation capability, we conducted an "Iso-Length" stress test on the HLE dataset with a batch size of 1, where both models were constrained to generate identical sequence lengths ranging from 128 to 2048 tokens. Figure 4(b) illustrates the throughput divergence. While the AR baseline maintains a relatively static throughput (∼10 tokens/sec) limited by memory bandwidth and serial dependency, MedVerse demonstrates superior parallel scalability. The performance gap widens significantly as the sequence length increases: at 2048 tokens, our system achieves a peak throughput of ∼17.1 tokens/sec, representing a +69.3% gain over the baseline. This confirms that MedVerse effectively converts the GPU's parallel compute capacity into valid token throughput, making it increasingly efficient for long-context medical reasoning tasks.

Stage / Metric	Wall-clock Time (%)
Planning Stage	39%
Execution Stage	61%
System Overhead (parsing & scheduling)	<0.01%
KV Fork/Join Cost (within Execution)	1.1%

Table 2: **Computational Cost Decomposition.** Wall-clock time breakdown for MedVerse.

Computational Cost Decomposition. We further profile the wall-clock time distribution across pipeline stages (Table 2) to isolate the sources of latency. The Planning Stage accounts for 39% of total wall-clock time, while the Execution Stage dominates at 61%. System-level parsing and scheduling overhead is negligible (<0.01%), and the combined KV Fork/Join cost represents only 1.1% of total wall-clock time, confirming that the observed speedups are not offset by infrastructure overhead. This is enabled by Radix Attention for $\mathcal{O}(1)$ Fork operations and zero-copy KV merges for Join primitives.

5.4 Structural Analysis

To characterize when our approach benefits, we profile MedVerse with Qwen-2.5-7B across DAG topologies. In cases of a single linear chain, which comprise only 3% of clinical cases, MedVerse maintains 1:1 latency parity with serial generation rather than degrading; the planning and parsing overhead is negligible at this scale. Significant efficiency gains are realized as DAG topologies becomes difficult (Table 3), shifting computational complexity from total token length to topological depth. This architectural alignment with non-linear medical reasoning ensures high reliability (up to 63.5% accuracy) without computational friction.

Topology	Prop.	Speedup	Acc. (%)
Single Linear Chain	3%	1.00×	–
Multi. Indep. Chains	58%	**1.40**×	54.3
Complex Intersecting	39%	**1.25**×	**56.7**

Table 3: **Efficiency and Structural Analysis.** MedVerse yields meaningful speedups across 97% of non-linear clinical cases while maintaining high accuracy.

5.5 Clinical Reliability Analysis

To evaluate clinical reliability, we conducted a rigorous analysis using GPT-5.2 as a physician-level judge (Zheng et al., 2023) across 50 cases from HLE (Biomed). We define five key metrics: Causal Validity (adherence to medical logic, 1–5 scale), Edge Accuracy (correctness of reasoning steps in the DAG, %), Average Logical Jumps (unsupported transitions, average count per case), Mean Score (overall clinical utility, 1–5 scale), and High-Risk Error (frequency of dangerous guideline contradictions, categorized by levels 1/3/5). As shown in Table 4, MedVerse demonstrates superior structural integrity; by anchoring reasoning branches in evidence via DAG-structured constraints, it significantly mitigates hallucinatory leaps. Notably, MedVerse reduces high-risk errors by 50% compared to the serial baseline (5.7% vs. 11.4%), while improving causal validity by 12.1% and edge accuracy by 15.4%. These results confirm that our Join-based verification effectively enforces logical consistency in high-stakes clinical decision-making.

Metric	Serial	MedVerse	Δ
Causal Validity ↑	1.82	**2.04**	+12.1%
Edge Accuracy ↑	35.8	**41.3**	+15.4%
Logical Jumps ↓	3.30	**2.46**	-25.5%
Mean Score ↑	2.00	**2.36**	+18.0%
High-Risk Error ↓	11.4	**5.7**	-50.0%

Table 4: **Clinical Reliability Evaluation** on 50 HLE (Biomed) cases. ↑ (↓) indicates higher (lower) is better.

5.6 Ablation Study

We conduct a targeted ablation to assess the necessity of linear-to-parallel hybrid reasoning by comparing MedVerse with a **Direct Petri Net** variant that directly generates topological structures without linear planning. As shown in Table 5, when evaluated on MedXpert (batch size 1), the Direct Petri Net variant exhibits a substantial accuracy drop and underperforms even the autoregressive baseline, indicating that standard LLMs struggle to construct sound execution graphs from scratch.

Model Variant	Linear	Parallel	Accuracy (%)	Latency (s)
Autoregressive	✓	×	18.4	5.1
Direct Petri Net	×	✓	17.4	4.5
MedVerse	✓	✓	**19.3**	4.0

Table 5: **Efficacy of Linear-to-Parallel Hybridization.** Ablation study on the MedXpert benchmark.

In contrast, MedVerse achieves both the highest accuracy (19.3%) and the lowest latency (4.0s), demonstrating that linear planning is essential for reliable graph construction and that its combination with parallel execution yields the best accuracy–efficiency trade-off.

We further disentangle architectural from data contributions in Appendix A.3: under identical MedVerse-14K data, the DAG-based attention and parallel inference yield a +2.4% gain over the standard autoregressive baseline (Table 8), confirming that the improvements are attributable to the architectural design rather than data exposure alone.

6 Conclusion

This work addresses the fundamental mismatch between sequential autoregressive decoding and the inherently parallel nature of clinical reasoning. By reformulating medical inference as a DAG-structured process grounded in Petri Net theory, MedVerse enables large language models to reason over multiple diagnostic hypotheses concurrently while preserving causal consistency. This paradigm alleviates key limitations of linear chain-of-thought reasoning in accuracy, efficiency, reliability, and interpretability, and offers a principled path toward structurally aligned medical reasoning systems suitable for real-world clinical decision support.

7 Ethical Considerations

This work focuses on methodological advances in structured and parallel reasoning for medical language models and does not involve new data collection from human subjects. All training and evaluation data are derived from existing, publicly available benchmarks or automatically generated synthetic reasoning trajectories. No personally identifiable information is used. While the proposed framework aims to improve reasoning accuracy, efficiency, and interpretability, it is not intended for direct clinical deployment without human oversight. Any real-world medical application should ensure appropriate clinical validation and adhere to established ethical and regulatory standards.

Limitations

While MedVerse demonstrates consistent gains in both reasoning accuracy and inference efficiency, several limitations merit discussion. First, although MedVerse reduces inference latency by shifting computational complexity from sequence length to topological depth, the achievable speedup depends on the inherent parallelism of the reasoning graph. For reasoning problems with predominantly linear dependency structures, the benefits of parallel execution may be limited. Automatically adapting execution strategies to different reasoning topologies remains an important direction for future work. Second, our evaluation focuses on offline benchmarks under controlled inference settings. While the proposed framework is designed with real-time clinical deployment in mind, additional studies are required to assess its behavior under interactive, user-in-the-loop scenarios, where user feedback, partial responses, or dynamic query refinement may influence the execution process.

Acknowledgement

This work is partially supported by NSF #2449442 and NIH R01AG085581, R01AG079291, R01AR083790 and P50HD103573. The Authors acknowledge the National Artificial Intelligence Research Resource (NAIRR) Pilot, NCSA DeltaAI and OpenAI API for contributing to this research result.

References

Hasan Md Tusfiqur Alam, Devansh Srivastav, Md Abdul Kadir, and Daniel Sonntag. 2025. Towards interpretable radiology report generation via concept bottlenecks using a multi-agentic rag. In *European Conference on Information Retrieval*, pages 201–209. Springer.

Judith L Bowen. 2006. Educational strategies to promote clinical diagnostic reasoning. *New England Journal of Medicine*, 355(21):2217–2225.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Hanjie Chen, Zhouxiang Fang, Yash Singla, and Mark Dredze. 2025. Benchmarking large language models on answering and explaining challenging medical questions. 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 3563–3599.

Junying Chen, Zhenyang Cai, Ke Ji, Xidong Wang, Wanlong Liu, Rongsheng Wang, Jianye Hou, and Benyou Wang. 2024. Huatuogpt-o1, towards medical complex reasoning with llms. *arXiv preprint arXiv:2412.18925*.

Yifu Ding, Wentao Jiang, Shunyu Liu, Yongcheng Jing, Jinyang Guo, Yingjie Wang, Jing Zhang, Zengmao Wang, Ziwei Liu, Bo Du, and 1 others. 2025. Dynamic parallel tree search for efficient llm reasoning. *arXiv preprint arXiv:2502.16235*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Arthur S Elstein, Lee S Shulman, and Sarah A Sprafka. 1978. *Medical problem solving: An analysis of clinical reasoning*. Harvard University Press.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Xiaoke Huang, Juncheng Wu, Hui Liu, Xianfeng Tang, and Yuyin Zhou. 2025a. m1: Unleash the potential of test-time scaling for medical reasoning with large language models. *arXiv preprint arXiv:2504.00869*.

Xiaoke Huang, Juncheng Wu, Hui Liu, Xianfeng Tang, and Yuyin Zhou. 2025b. Medvlthinker: Simple baselines for multimodal medical reasoning. *arXiv preprint arXiv:2508.02669*.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. 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 2567–2577.

Jerome P Kassirer and G Anthony Gorry. 1978. Clinical problem solving: a behavioral analysis. *Annals of Internal Medicine*, 89(2):245–255.

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.

Yuxiang Lai, Jike Zhong, Ming Li, Shitian Zhao, Yuheng Li, Konstantinos Psounis, and Xiaofeng Yang. 2025. Med-r1: Reinforcement learning for generalizable medical reasoning in vision-language models. *arXiv preprint arXiv:2503.13939*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. 2023a. Llava-med: Training a large language-and-vision assistant for biomedicine in one day. *Advances in Neural Information Processing Systems*, 36:28541–28564.

Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. 2023b. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6).

Thang Luong, Edward Lockhart, and 1 others. 2025. Advanced version of gemini with deep think officially achieves gold-medal standard at the international mathematical olympiad. *Google DeepMind Blog*, 1.

Michael Moor, Oishi Banerjee, Zahra Shakeri Hossein Abad, Harlan M Krumholz, Jure Leskovec, Eric J Topol, and Pranav Rajpurkar. 2023a. Foundation models for generalist medical artificial intelligence. *Nature*, 616(7956):259–265.

Michael Moor, Qian Huang, Shirley Wu, Michihiro Yasunaga, Yash Dalmia, Jure Leskovec, Cyril Zakka, Eduardo Pontes Reis, and Pranav Rajpurkar. 2023b. Med-flamingo: a multimodal medical few-shot learner. In *Machine Learning for Health (ML4H)*, pages 353–367. PMLR.

Vishwesh Nath, Wenqi Li, Dong Yang, Andriy Myronenko, Mingxin Zheng, Yao Lu, Zhijian Liu, Hongxu Yin, Yee Man Law, Yucheng Tang, and 1 others. 2025. Vila-m3: Enhancing vision-language models with medical expert knowledge. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 14788–14798.

Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. 2023. Capabilities of gpt-4 on medical challenge problems. *arXiv preprint arXiv:2303.13375*.

OpenAI. 2024. Learning to reason with LLMs. Accessed: 2025-01-04.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pages 248–260. PMLR.

Jiayi Pan, Xiuyu Li, Long Lian, Charlie Snell, Yifei Zhou, Adam Yala, Trevor Darrell, Kurt Keutzer, and Alane Suhr. 2025. Learning adaptive parallel reasoning with language models. *arXiv preprint arXiv:2504.15466*.

Carl Adam Petri. 1962. Kommunikation mit automaten.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, and 1 others. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, and 1 others. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Qwen Team and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3).

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature medicine*, 29(8):1930–1940.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Ziqi Wang, Boye Niu, Zipeng Gao, Zhi Zheng, Tong Xu, Linghui Meng, Zhongli Li, Jing Liu, Yilong Chen, Chen Zhu, and 1 others. 2025. A survey on parallel reasoning. *arXiv preprint arXiv:2510.12164*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Chaoyi Wu, Xiaoman Zhang, Ya Zhang, Yanfeng Wang, and Weidi Xie. 2023. Pmc-llama: Further finetuning llama on medical papers. *arXiv preprint arXiv:2304.14454*, 2(5):6.

Juncheng Wu, Wenlong Deng, Xingxuan Li, Sheng Liu, Taomian Mi, Yifan Peng, Ziyang Xu, Yi Liu, Hyunjin Cho, Chang-In Choi, and 1 others. 2025. Medreason: Eliciting factual medical reasoning steps in llms via knowledge graphs. *arXiv preprint arXiv:2504.00993*.

Peng Xia, Ze Chen, Juanxi Tian, Yangrui Gong, Ruibo Hou, Yue Xu, Zhenbang Wu, Zhiyuan Fan, Yiyang Zhou, Kangyu Zhu, and 1 others. 2024a. Cares: A comprehensive benchmark of trustworthiness in medical vision language models. *Advances in Neural Information Processing Systems*, 37:140334–140365.

Peng Xia, Jinglu Wang, Yibo Peng, Kaide Zeng, Xian Wu, Xiangru Tang, Hongtu Zhu, Yun Li, Shujie Liu, Yan Lu, and Huaxiu Yao. 2025a. Mmedagent-rl: Optimizing multi-agent collaboration for multimodal medical reasoning. *arXiv preprint arXiv:2506.00555*.

Peng Xia, Kangyu Zhu, Haoran Li, Tianze Wang, Weijia Shi, Sheng Wang, Linjun Zhang, James Zou, and Huaxiu Yao. 2025b. Mmed-rag: Versatile multimodal rag system for medical vision language models. In *The Thirteen International Conference on Learning Representations*.

Peng Xia, Kangyu Zhu, Haoran Li, Hongtu Zhu, Yun Li, Gang Li, Linjun Zhang, and Huaxiu Yao. 2024b. Rule: Reliable multimodal rag for factuality in medical vision language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1081–1093.

Xinyu Yang, Yuwei An, Hongyi Liu, Tianqi Chen, and Beidi Chen. 2025. Multiverse: Your language models secretly decide how to parallelize and merge generation. *arXiv preprint arXiv:2506.09991*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Hongbo Zhang, Junying Chen, Feng Jiang, Fei Yu, Zhihong Chen, Guiming Chen, Jianquan Li, Xiangbo Wu, Zhang Zhiyi, Qingying Xiao, and 1 others. 2023. Huatuogpt, towards taming language model to be a doctor. In *Findings of the association for computational linguistics: EMNLP 2023*, pages 10859–10885.

Wenchuan Zhang, Jingru Guo, Hengzhe Zhang, Penghao Zhang, Jie Chen, Shuwan Zhang, Zhang Zhang, Yuhao Yi, and Hong Bu. 2025. Patho-agenticrag: Towards multimodal agentic retrieval-augmented generation for pathology vlms via reinforcement learning. *arXiv preprint arXiv:2508.02258*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, and 1 others. 2024. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583.

Kangyu Zhu, Peng Xia, Yun Li, Hongtu Zhu, Sheng Wang, and Huaxiu Yao. 2025. Mmedpo: Aligning medical vision-language models with clinical-aware multimodal preference optimization. *Forty-Second International Conference on Machine Learning*.

Yuxin Zuo, Shang Qu, Yifei Li, Zhangren Chen, Xuekai Zhu, Ermo Hua, Kaiyan Zhang, Ning Ding, and Bowen Zhou. 2025. Medxpertqa: Benchmarking expert-level medical reasoning and understanding. *arXiv preprint arXiv:2501.18362*.

A Additional Analysis

A.1 Effect of Training Data Scale

Samples	1k	2k	5k	8k	14k
Subset Ratio	∼7%	∼14%	∼36%	∼57%	100%
Average Accuracy	29.2%	32.5%	37.5%	38.7%	**39.2%**

Table 6: **Scalability Analysis.** The monotonic improvement in average accuracy across all benchmarks confirms the effectiveness of our data scaling strategy.

We investigate the scalability of our approach by fine-tuning the base model on varying subsets of the MedVerse-14K dataset (ranging from 1k to 14k samples). As presented in Table 6, we observe a clear monotonic positive correlation between dataset size and the average reasoning accuracy across all evaluation benchmarks. Notably, the model exhibits exceptional data efficiency: with only 5k samples (∼36% of the total data), it achieves an accuracy of 37.5%, recovering over 95% of the peak performance. Furthermore, the performance does not plateau even at the 14k mark. This validates the high quality of our synthesized data and suggests that further scaling could yield even greater improvements.

A.2 Scalability Across Model Sizes

To assess the scalability of MedVerse beyond 7B and 8B scales, we evaluate MedVerse fine-tuned on the Qwen2.5-14B-Instruct backbone. As shown in Table 7, **MedVerse-14B** achieves 44.74% average accuracy, representing a **+18.0%** relative gain over the official 14B baseline—exceeding the relative gain observed at the 7B scale (+14.2%). This demonstrates that our DAG-structured inductive bias becomes increasingly effective as parameters increase. Crucially, MedVerse maintains a consistent 1.32× inference latency speedup at the 14B scale. As it is implemented via topology-aware attention masking and adaptive position indices, the framework remains compatible with larger Transformers and MoE architectures.

Model	HLE	MB-op4	MB-op5	MedQA	MedXpert	Avg.
Qwen2.5-7B (Base)	18.4	45.8	39.6	56.2	12.3	34.46
MedVerse-7B	19.6	55.2	48.0	58.6	15.3	39.34
Qwen2.5-14B (Base)	10.7	54.2	49.0	61.7	13.9	37.91
MedVerse-14B	26.2	62.3	52.0	64.0	19.2	**44.74**

Table 7: **Scaling Performance Analysis.** MedVerse-14B achieves +18.0% relative gain, exceeding the 7B-scale gain (+14.2%), confirming that DAG-structured reasoning scales effectively.

A.3 Analysis of Training Strategies and Inference Modes

To further investigate the source of MedVerse's performance gains, we conducted a comprehensive ablation study decoupling the training strategies (standard autoregressive vs. MedVerse attention) from the inference execution mode (serial vs. parallel). We evaluated four configurations on the Qwen2.5-7B-Instruct backbone across our benchmarks. The aggregated results are summarized in Table 8.

The four configurations are defined as follows.

Auto-Ser: Standard autoregressive training executed with a standard serial engine (baseline).

Auto-Par: Standard autoregressive training executed with our DAG-based parallel engine.

Mask-Ser: MedVerse topology-aware training (masked attention) executed serially.

Mask-Par: MedVerse topology-aware training executed with our DAG-based parallel engine.

Metric	Auto-Ser	Auto-Par	Mask-Ser	Mask-Par (Ours)
Average Accuracy (%)	36.9	37.9	38.6	**39.3**

Table 8: Ablation study on Training Strategies and Inference Modes. We report the average accuracy across five medical reasoning benchmarks (HLE, MedBullets-op4/op5, MedQA, MedXpert).

Discussion. The results demonstrate a strong synergistic effect, with Mask-Par achieving the highest accuracy (39.34%, +2.44% over baseline). This improvement highlights two critical insights. First, the superiority of Mask-Ser over the baseline (+1.66%) confirms that topology-aware masking enhances learning; by "hiding" irrelevant parallel branches, the mechanism prevents reliance on spurious positional correlations and forces the model to focus on genuine causal dependencies. Second, regarding topological alignment, the consistent gains from parallel structures in both training and inference indicate that medical reasoning inherently follows a DAG topology rather than a linear chain. MedVerse succeeds precisely by aligning the computational framework with this non-linear cognitive structure.

B Detailed MedVerse Curator Pipeline

This appendix provides a detailed description of the MedVerse Curator, including phase-wise procedures for knowledge grounding, topological planning, structural synthesis, and data verification.

Phase 1: Knowledge-Grounded Path Initialization. The pipeline begins by anchoring the reasoning in established medical knowledge, leveraging the retrieval methodology from MedReason (Steps 1–3):

(i) **Knowledge Retrieval:** We first retrieve potential reasoning paths connecting the question entities to answer candidates from a large-scale medical Knowledge Graph (KG).

(ii) **Entity Mapping:** We perform medical entity extraction to map unstructured query terms to standardized KG nodes.

(iii) **Path Pruning:** We search for and prune irrelevant branches to isolate the original, raw reasoning paths.

This phase provides a "ground truth" skeleton, ensuring the subsequent generation is factually grounded rather than hallucinated.

Phase 2: Topological Planning and Filtering. We then transform these linear skeletons into the MedVerse architecture (Steps 4–5). We employ a specialized prompt to **filter and edit** the raw paths, removing redundancy and ensuring logical coherence. Crucially, we perform a **DAG Validity Check**: the refined path is analyzed to ensure it forms a valid Directed Acyclic Graph. If the dependencies form a cycle, the path is rejected or re-routed.

Phase 3: Structural Synthesis and Refinement. This core phase generates the XML-structured data (Steps 6–8):

- **Plan Generation (`<Plan>`):** We iteratively decompose the verified path, utilizing regex-based formatting to outline the Petri Net structure, defining transitions and explicit dependency lists.
- **Parallel Execution (`<Execution>`):** The teacher LLM generates the "transient step" content for each transition.
- **Iterative Refinement:** We implement a **Refinement Module** to knit these independent steps into a cohesive narrative. This module deduplicates overlapping logic across parallel branches, removes non-contributory details, and smooths the transition flow to ensure the reasoning naturally bridges the gap from the problem description to the final entity.
- **Conclusion Synthesis:** Finally, conditioned on the refined execution trajectory, the model generates the `<Conclusion>`, providing the final answer and a holistic explanation.

Phase 4: Dual-Layer Verification Loop. To guarantee data quality, we append two supplementary validation stages:

(a) **Syntax Verification:** Ensures strict adherence to the XML schema and Petri Net definition (e.g., matching `<Step>` indices to `<Outline>` plans).

(b) **Logic & Completeness Evaluation:** An evaluator model assesses the reasoning chain for logical gaps and verifies that the conclusion correctly addresses the user goal.

Data failing either check triggers an iterative regeneration loop until all criteria are met.

C Prompting Protocol for the MedVerse Curator

In this section, we present the complete five-stage prompting protocol used by the MedVerse Curator to construct the MedVerse-14K dataset, powered by the GPT-5.1 model accessed via the ChatGPT API. The goal of this protocol is to systematically transform question–answer pairs into structured, parallel medical reasoning trajectories suitable for topology-aware execution.

The protocol is implemented as an offline data construction pipeline consisting of five sequential phases. Phase 1 adopts the knowledge-grounded retrieval procedure proposed in MedReason to initialize medically valid reasoning paths; as this phase directly reuses an existing method without modification, we do not reproduce the original prompts. Phases 2–4 introduce new structured transformations that progressively induce parallel reasoning structure, repopulate detailed clinical content, and enforce execution-ready constraints. The full specifications and prompts for Phases 2–4 are listed below.

D Use of Large Language Models

We utilized large language models (LLMs) exclusively for linguistic refinement and as coding assistants for routine programming tasks. These tools played no role in the study's conceptualization, experimental design, data analysis, or the interpretation of result. The authors retain full responsibility for the validity and originality of the scientific content.

Phase 2: Reasoning Chain Filtering Template

SYSTEM PROMPT

You are a strict reasoning chain filter. Given a question, a list of candidate reasoning chains (`original_reasoning`), and the correct answer, select only the reasoning chains that are directly relevant for deriving the answer from the question.

Filtering Rules (Follow All Exactly):

1) **Relevance:** Keep only chains that directly or critically contribute to deriving the answer from the question. Discard any chain that is unrelated or unnecessary for reaching the answer.
2) **Consistency:** Remove chains that contradict the facts stated in the question or that lead to conclusions conflicting with the answer.
3) **Duplicate Removal:** If multiple chains are textually identical, keep only the first occurrence.
4) **Order & Priority:** Preserve the original order of appearance in `original_reasoning`. If more than 10 chains remain, output the 10 most useful ones for deriving the answer (strongest/direct connection).
5) **Text Integrity:** Do not modify any retained reasoning chain text. Each chain must remain exactly identical to its original text (from `'A->B->C->...'` to the end). Only reassign new indices starting from 1 in ascending order.
6) **Empty Case:** If no chains satisfy the rules, output nothing (no text, no comment).

USER INPUT

Input:

- **question:** {question}
- **answer:** {answer}
- **original_reasoning** (each line formatted as `"<index>: A->B->C->..."`): {original_reasoning}

Output:

Return only the filtered reasoning chains in this exact format and nothing else:

```
1: reasoning chain text (identical to original)
2: reasoning chain text (identical to original)
...
(Up to 10 lines total.)
```

Phase 2: Reasoning Chain Refinement Template

SYSTEM PROMPT

You are an expert in the medical domain.

Goal: Generate reasoning chains where the **Start Node** is strongly correlated with a key entity in the Question, and the **End Node** is strongly correlated with the Answer entity.

STRONG PRIORITY ON USING PROVIDED PATHS:

- Maximize reuse of entities and links that appear in the provided Paths.
- Prefer chains composed entirely of nodes from the Paths.
- If you reuse nodes from the Paths, keep their strings EXACTLY as written (same casing, no edits).
- Select Start/End nodes from the provided Paths that have the highest semantic or clinical correlation to the Question/Answer context.

STRICT OUTPUT RULES:

1) **Format:** Each line: `'<index>: A->B->C->...'` (Index starts at 1, exactly one space after colon).
2) **Delimiter:** Use `'->'` as the ONLY delimiter with no spaces around it.
3) **Start Node:** Must have a STRONG CORRELATION (semantic or clinical) to a key entity in the Question (does NOT need to be an exact string match).
4) **Final Node:** Must have a STRONG CORRELATION to the Answer entity (does NOT need to be the exact Answer string).
5) **Validity:** Each link A→B must be a medically valid causal/inferential relation.
6) **Cleanliness:** No headers, no explanations, no extra text. Do not rewrite/rename nodes taken from the Paths.
7) **Quantity:** Output up to 6 chains; output nothing if no valid chain can be formed.
8) **Preference:** If multiple valid options exist, prefer chains that (a) maximize coverage of nodes from the provided Paths, and (b) require zero new nodes (or at most one new medically sound bridge).
9) **Diversity:** Avoid producing only a single reasoning chain unless only one medically valid path exists; whenever possible, output two or more distinct valid reasoning chains.

USER INPUT

- **Question:** {question}
- **Answer:** {answer}
- **Paths (filtered reasoning paths to reuse):** {filter_reasoning_path}

Output:

```
1: A->B->...
2: A->B->...
...
```

Phase 2: Reasoning Chain Editing Template

SYSTEM PROMPT

You are a medical-domain reasoning chain editor.

Edit policy (hard constraints):

- **Identity-by-default:** If a chain is already complete and logically sound, output it UNCHANGED.
- **Only-if-incomplete:** Modify a chain ONLY when it is logically incomplete between Question-entity and Answer-entity/synonym.
- **Preserve-original-entities:** Do NOT alter, delete, paraphrase, or reorder ANY existing entities or links.
- **Insert-only-new-bridge:** When a fix is necessary, INSERT the FEWEST possible concise medical entities as bridges; do not modify existing tokens. Prefer ≤ 2 new entities per chain.
- **Medical validity:** Each hop A$\rightarrow$B must be a clinically valid causal/inferential relation.
- **Uncertainty:** If uncertain whether a change is required, leave the chain UNCHANGED.

USER INPUT

Requirements:

1) Only add new entities to chains that are logically incomplete; keep ALL original entities exactly as given (no edits, no reordering, no deletions).
2) Use `'->'` only; no spaces around it. Output one line per chain as `'<index>: A->B->C->...'`; indices start at 1, increment by 1, and have exactly one space after `':'`.

Input Data:

- **Question:** {question}
- **Answer:** {answer}
- **new_reasoning_path:** {new_reasoning_path}

Output:

```
1: A->B->...
2: A->B->...
...
```

Phase 3: Atomic Reasoning Step Template

SYSTEM PROMPT

You are an expert in the medical domain. Your task is to perform **Chain-of-Thought (CoT) reasoning for a single step independently**, strictly based on prior dependency results and current step keywords.

Core Task: Reason through **only the current step**. Rely solely on: 1) The CoT results from the directly dependent previous steps. 2) The entity keywords specific to the current step.

Strict Constraints (Do NOT):

- Do not explain the overall goal or speculate about future steps.
- Do not explicitly mention "dependencies" or reference previous steps by name (e.g., "based on step 1").
- Do not introduce extraneous definitions or general knowledge unrelated to the specific structural logic.
- Do not state functions or importance; stick to factual, anatomy-based structural logic.

Mandatory Requirements:

- **Conciseness:** Output only a single, short CoT paragraph.
- **Entity Detail:** All entities mentioned in the current step AND its dependencies **MUST** have their factual details explicitly included (e.g., structural, anatomical, or pathological attributes).
- **completeness:** The reasoning is considered incorrect if any entity detail is omitted or vaguely referenced.

USER INPUT

Input Data:

1. **Goal:** {goal}
2. **Plan** (Steps & Dependencies): {plan}
3. **Executed Steps** (CoT results from dependencies only): {executed_step}
4. **Current Step** (Focus of this reasoning): {current_step}

Output Instruction: CoT Paragraph: (Provide only one short paragraph of factual reasoning. Do not reference the answer or future steps.)

Phase 3: Reasoning Refinement Template

SYSTEM PROMPT

You are an expert in extracting concise, non-redundant, factual reasoning from multi-step medical analyses.
Input Context: You will be given: 1) A **goal**, describing the final medical question. 2) A series of **step-wise reasoning outputs**, where each step includes a label (`Entity A -> Entity B`) and a detailed reasoning paragraph.
Task: Eliminate any redundant content that has already appeared in previous steps. Keep only factual details necessary for reasoning toward the goal.
Strict Constraints:

- **No Redundancy:** Remove information repeated from previous steps.
- **Facts Only:** Contain only medical facts; exclude definitions, purpose, significance, or usefulness.
- **Logic Preservation:** Must retain the logical relationship between Entity A and Entity B.
- **Format:** Return the original step label followed by your revised concise reasoning.
- **No Meta-Content:** Do not include explanations, summaries, or overall conclusions.

USER INPUT

- **Goal:** {goal}
- **Stepwise Reasoning:** {multistep_reasoning}

Phase 3: Conclusion Synthesis Template

SYSTEM PROMPT

You are an expert in medical reasoning. You will be given three inputs: 1) Reasoning Paragraphs (a sequence of `'Transient Step N: ...'` logical steps), 2) Question (a medical multiple-choice question), 3) Options (possible answers).
Your Task:

- Use **ONLY** the Reasoning Paragraphs to determine the correct answer.
- First, output the final answer to the Question.
- Then, output one concise paragraph explaining why this is the correct answer, referencing the relevant steps.
- **Constraint:** Do not add external knowledge.

Output Format:

```
Explanation: <one paragraph justification>
Answer: <the correct option>
```

USER INPUT

- **Reasoning Paragraphs:** {total_final}
- **Question:** {question}
- **Options:** {option}

Phase 4: Answer & Logic Verification Template

SYSTEM PROMPT

You are an expert evaluator of medical reasoning consistency.
Input Context: You will be given: 1) A **goal** (the correct answer). 2) A **question** and its **options**. 3) A **conclusion** (final answer + explanation). 4) A set of **reasoning steps** (the chain of thought).
Your Tasks:

1. **Answer Verification:** Verify whether the conclusion's final answer matches the correct answer specified in the goal.
2. **Logic Verification:** Verify whether the explanation in the conclusion is logically consistent with the reasoning steps.
   - The explanation must be directly derivable from the given reasoning steps.
   - It must **not** rely on external facts or background knowledge not present in the reasoning steps.

Output Requirements:

- Output **"Consistent"** if AND ONLY IF: (a) the answer matches the goal AND (b) the explanation is logically derived solely from the steps.
- Otherwise, output **"Inconsistent"**.
- Do not output anything else.

USER INPUT

- **Goal (correct answer):** {goal}
- **Question:** {question}
- **Options:** {options}
- **Conclusion (explanation + answer):** {conclusion}
- **Reasoning Steps:** {reasoning_steps}