GuideTree: Guideline-Induced Review Trees for Long Medical Records

Chengze Ge, Ruiqing Zhang, Yining Wang, Shengping Liu,
Jiaen Liang, Wei Huang, Weihua Chen[†]
Unisound AI Technology Co., Ltd.
gechengze@unisound.com, chenweihua@unisound.com

Abstract

Reviewing medical records for clinical and insurance decisions must handle long, heterogeneous documents while producing consistent, traceable, guideline-compliant outcomes under strict latency and cost constraints. We propose **GuideTree**, which compiles textual guidelines into a *fixed* review tree of evidence-grounded verification primitives. GuideTree uses short per-document summaries *only* for routing each check to a minimal set of document types and candidates; final verification always reads full document text and returns structured evidence. The tree is induced offline via a cost-aware split-and-prune search and updated safely through regression-tested, versioned patches. Across 1,000 cases from four industrial review scenarios and four LLM backbones, GuideTree achieves 84.5–92.8 Macro-F1, outperforming the strongest non-expert baselines by 3.3–7.6 points and matching ExpertTree within 0.2–0.6 points (avg. 0.38). On chronic disease with Qwen3-235B-A22B-Instruct, GuideTree reduces average I/O volume to 74K input+output characters (-82% vs. long-context prompting) and average latency to 22s (-83% vs. long-context prompting), while reaching 99% decision consistency over $K=5$ reruns. Code is available at https://github.com/gechengze/guidetree.

1 Introduction

Medical record review underpins clinical operations and insurance decisions, including documentation quality control, coding verification, guideline compliance, and claim adjudication. Reviewers must aggregate evidence scattered across long, heterogeneous records (e.g., notes, lab/imaging reports, prescriptions, procedures, and claim materials) while producing decisions that are consistent, traceable, and reproducible under strict latency and cost constraints.

[†] Corresponding author.

LLMs enable automation, but long-record review remains difficult in deployment. Long-context inference is expensive and often degrades as inputs grow; retrieval-based reduction can miss critical evidence dispersed across document types. Agentic pipelines improve flexibility by iterating planning, retrieval, and reasoning, yet they are often sequential and costly, and their intermediate choices (e.g., ordering and exploratory retrieval) can drift across runs, harming reproducibility and auditability. In audit-oriented settings, such instability is a product risk: the same record and policy should yield the same decision and evidence path, and failures must be localizable to specific checks.

Deterministic expert-authored review trees are therefore often more reliable and efficient than generic agent baselines because they encode stable, evidence-grounded checks. However, expert trees are expensive to build and maintain as scenarios and corner cases expand.

We propose **GuideTree**, which compiles textual guidelines into executable, tree-structured workflows for long medical records. GuideTree maps guideline statements to primitive conditions and binds them to reusable verification primitives with explicit parameters. For efficiency, it uses short per-document summaries *only* for routing each check to a minimal set of document types and candidate sources; final verification always reads full document text and returns structured evidence. To ensure stable deployment behavior, GuideTree induces a *fixed* tree offline via a cost-aware split-and-prune procedure that trades off decision utility, execution cost, and structural complexity, enabling early stopping and parallelizable leaf checks. Finally, GuideTree supports controlled post-deployment evolution through failure localization and regression-tested, versioned updates. Figure 1 summarizes the compilation and execution pipeline.

Our contributions are:

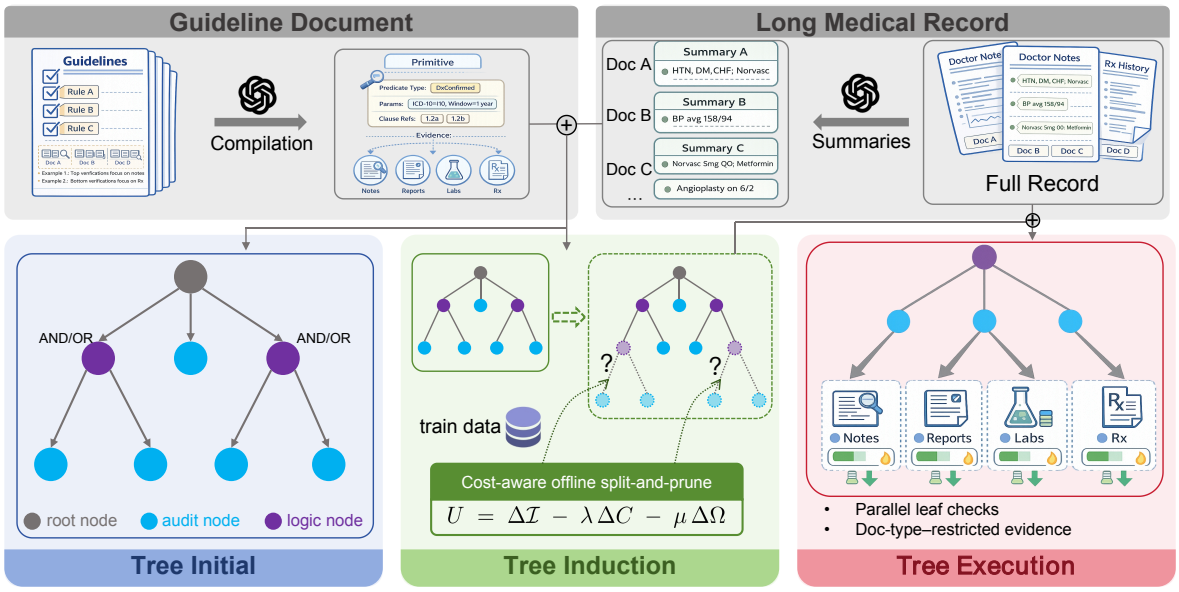

Figure 1: Overview of **GuideTree**. We compile guidelines into a fixed review tree of verification primitives, using summaries only for routing and full-text verification for evidence; the tree is induced offline and executed deterministically with parallel leaves.

- We propose **GuideTree**, a guideline-induced framework that compiles textual policies into executable, auditable review trees for long, heterogeneous medical records.
- We bind guideline conditions to reusable **verification primitives** with explicit parameters, and route each check through **minimal interfaces** using lightweight document summaries (routing only; full text for verification).
- We introduce an **offline cost-aware split-and-prune** procedure to induce a *fixed*, deterministic deployment tree, with **regression-tested, versioned updates** for safe post-deployment refinement.

2 Related Work

Long-context modeling and retrieval augmentation. Long-document processing is supported by efficient-attention or memory-based architectures such as Transformer-XL (Dai et al., 2019), Longformer/LED (Beltagy et al., 2020), BigBird (Zaheer et al., 2020), Reformer (Kitaev et al., 2020), and Linformer (Wang et al., 2020), yet performance can degrade when evidence is sparse and dispersed (Liu et al., 2024). Retrieval-augmented generation (RAG) reduces inputs by conditioning on retrieved evidence (Lewis et al., 2020), e.g., DPR (Karpukhin et al., 2020), ColBERT (Khattab and Zaharia, 2020), and FiD-style aggregation (Izacard and Grave, 2021); ensuring complete coverage under strict budgets in heterogeneous medical records remains challenging.

Agentic, tool-augmented, and multi-agent review. Agent frameworks interleave planning, retrieval, and reasoning (Yao et al., 2022; Wang et al., 2023), and tool-use paradigms emphasize modular routing and learned invocation (Ahn et al., 2022; Schick et al., 2023). Tree-structured search and sampling-based reasoning can improve success but introduce path variability that harms reproducibility (Yao et al., 2023; Wang et al., 2022). Related lines of work study self-reflection and feedback-driven refinement across episodes for improving agent behavior (Shinn et al., 2023). Multi-agent designs coordinate role-specialized agents via structured aggregation (Yu et al., 2025; Zhang et al., 2024), and tree-structured review explicitly organizes evidence checks for audit-oriented long-document review (Chang et al., 2025).

3 Method

3.1 Overview

Given a medical record $\mathcal{R}$ (a set of heterogeneous documents) and a scenario guideline document G (Appendix C), we output a binary decision $\hat{y} \in \{0, 1\}$ together with auditable evidence $\hat{\mathcal{E}}$. We compile G into a tree workflow T: leaves are *ver-*

ification primitives that read selected documents and return $(b_\ell, \mathcal{E}_\ell)$, while internal nodes deterministically aggregate booleans (e.g., $\wedge, \vee, \neg$). Evidence is the union of visited leaves, yielding reproducible evidence paths.

For efficiency on long records, we first generate short per-document summaries and use them *only* for routing (choosing relevant document types and a small candidate set). Final verification always reads full document text and returns structured evidence. The deployment tree is induced *offline* (Section 3.3) and then executed with a fixed, deterministic evaluation order; independent leaves can be parallelized, so latency is dominated by the slowest visited leaf. Prompt templates are listed in Appendix D (English translations; the original Chinese templates are used in deployment).

3.2 Document Summarization

Each document d_i (type t_i, metadata $\mathbf{m}_i$) is summarized into a strict-budget cue s_i (e.g., ≤ 50 characters) capturing key diagnoses, tests, treatments, and events. We form $\mathcal{S}_{\text{all}} = \{(t_i, s_i)\}_{i=1}^{n}$ and use it only to (i) select a minimal set of document types and (ii) surface a few candidate documents per primitive. OCR is treated as preprocessing; we do not study multimodal/layout or OCR accuracy. The summarization prompt is provided in Appendix D.1.

3.3 Tree Construction

A composite guideline condition can be executed monolithically, but splitting it into subconditions enables early stopping and exposes parallel leaf checks. GuideTree therefore learns a *deployable workflow* offline: the tree structure, a deterministic evaluation order, and per-primitive routing interfaces, driven by labeled cases and execution traces (primitive outcomes, retrieved candidates, and I/O chars + latency cost).

Primitive binding and minimal routing interfaces. We normalize G into a boolean formula over primitive conditions, where each condition is the smallest checkable requirement. Each condition is bound to a reusable primitive $p \in \mathcal{P}$ with scenario parameters. To keep execution efficient, each primitive has a minimal routing interface $(\mathcal{D}_p, \mathcal{K}_p)$: a small set of document types $\mathcal{D}_p \subseteq \mathcal{T}$ and summary-level cues $\mathcal{K}_p$ (keywords/regex or embedding queries over $\mathcal{S}_{\text{all}}$) to select a few candidate documents. Summaries are used only for routing; the primitive verifies by reading full text and returns evidence snippets in $\mathcal{E}$. Scenario parameters are instantiated directly from guideline clauses (e.g., ICD ranges, numeric thresholds, time windows, and required evidence types), enabling optional trace-back from each primitive and evidence item to clause IDs in G.

Executable leaves, not free-form judgments. Leaves are not free-form LLM judgments. Each leaf is a scenario-instantiated yes/no executable check with document-type restrictions, explicit parameters, and a strict structured output schema. Summaries are used only for routing; the final leaf decision is made only after reading the selected full-text documents and returning `satisfied`, `evidence`, and `reasoning`.

Worked example. For example, in the chronic-disease scenario, the root decision for diabetes eligibility is $\mathrm{OR}(\text{Type 1 diabetes}, \text{Type 2 diabetes})$. The Type 2 diabetes branch is an AND over four checks: (1) confirmed T2D diagnosis at a secondary-or-above hospital (ICD category E11); (2) glycemic evidence, e.g., random plasma glucose $\geq$ 11.1 mmol/L, fasting plasma glucose $\geq$ 7.0 mmol/L on at least three occasions, OGTT 2h $\geq$ 11.1 mmol/L, or HbA1c $\geq 6.5\%$; (3) history of at least one year; and (4) at least one complication branch. A diabetic neuropathy branch can be compiled into two executable leaves: a diagnosis leaf restricted to discharge-related documents, checking for "diabetic neuropathy" or ICD E11.4, and a test-result leaf restricted to examination reports, checking for required findings such as abnormal EMG/nerve conduction or abnormal quantitative sensory testing. Internal nodes then aggregate leaf outputs deterministically.

Cost-aware offline split-and-prune. We induce a *fixed* deployment tree via a controlled search over guideline-consistent splits: for a composite node, a candidate split replaces it with its immediate children under the top-level operator, preserving interpretability and limiting the search space. Each split is scored by a utility that trades off decision benefit against execution cost and structural growth:

$$U \;=\; \Delta\mathcal{I} \;-\; \lambda\,\Delta C \;-\; \mu\,\Delta\Omega, \qquad (1)$$

where $\Delta\mathcal{I}$ is uncertainty reduction estimated from empirical child outcomes on labeled data, ΔC is

Table 1: Macro-F1 (%) on four review scenarios (CI/CD/Pre/Dx) across four LLM backbones. Methods include LC, RAG, ReAct, Plan-and-Execute, CodeAct, CoA, ToA, TreeReview, ExpertTree, and GuideTree. Higher is better.

Method	Qwen3-32B				Qwen3-235B-A22B-Instruct				Qwen3-235B-A22B-Thinking				DeepSeek-R1-671B			
	CI	CD	Pre	Dx	CI	CD	Pre	Dx	CI	CD	Pre	Dx	CI	CD	Pre	Dx
LC	54.2	56.1	48.1	51.2	54.9	56.2	50.5	55.0	55.1	57.5	51.8	53.9	55.4	59.3	52.1	55.0
RAG	62.8	66.5	59.3	61.9	66.2	66.7	60.2	63.4	65.3	68.8	61.5	63.0	66.3	66.8	60.5	64.6
ReAct	84.2	85.2	79.6	82.6	85.4	87.0	79.6	84.0	83.4	86.1	81.6	84.8	86.6	87.0	80.6	83.5
Plan-and-Execute	83.6	84.6	78.8	81.9	83.8	85.1	80.0	84.0	84.4	85.2	79.9	83.1	85.7	86.6	79.8	83.8
CodeAct	68.4	70.3	63.9	66.3	69.8	71.1	65.9	68.4	70.7	72.4	65.9	68.1	70.8	72.2	65.4	69.3
CoA	74.2	75.3	70.1	73.0	74.9	75.1	69.3	73.4	74.7	76.7	69.7	74.0	75.4	76.6	71.2	73.3
ToA	74.0	73.8	68.3	72.9	75.8	74.4	69.0	72.6	74.9	77.3	69.2	73.0	75.2	76.2	70.1	73.6
TreeReview	64.1	64.8	60.2	63.3	64.5	66.7	59.6	64.4	64.1	65.8	58.7	65.3	66.4	66.9	59.4	64.1
ExpertTree	**90.4**	**92.9**	**85.0**	**90.5**	**90.9**	**92.4**	**85.4**	**90.2**	**91.2**	**93.1**	**85.2**	**89.6**	**91.6**	**93.1**	**86.0**	**90.6**
GuideTree	90.2	92.5	84.5	90.2	90.7	92.0	84.8	89.9	90.9	92.8	84.9	89.1	91.2	92.5	85.5	90.3

the expected I/O chars + latency change using measured primitive costs and stop probabilities under $\wedge/\vee$, and $\Delta\Omega$ penalizes added nodes/depth. We apply greedy best-first split-and-prune: iteratively take the best positive-U split, update local statistics, and stop when no split helps or a budget is reached. For each operator, we fix a deterministic evaluation order that prioritizes cheap, high-stop-probability checks, improving efficiency and run-to-run consistency. We tune (λ, μ) on the dev set per scenario with a small grid under deployment tolerances, and then fix them for all test runs and backbones; Appendix B reports the chosen values and a sensitivity study.

Failure-driven refinement and regression-tested updates. Failures are localized by tracing the executed path and returned evidence, then patched *offline* with minimal changes: expand $(\mathcal{D}_p, \mathcal{K}_p)$ only when relevant evidence exists but was missed by routing; otherwise adjust local split/merge or ordering when decomposition harms the cost–quality trade-off. Each patch is validated on a held-out regression suite under cost/accuracy tolerances, producing versioned, deployment-ready workflows.

4 Experiments

4.1 Setup

We evaluate GuideTree on four industrial medical record review scenarios: CI (Critical Illness) for major-disease policy eligibility, CD (Chronic Disease) for chronic eligibility/management rules, Pre (Pre-existing) for time-windowed pre-existing condition decisions, and Dx (Diagnosis Review) for diagnosis support and/or coding/mapping. Each case is a heterogeneous multi-document record (e.g., notes, lab/imaging reports, prescriptions, procedures, and claim materials). The system outputs a binary decision with structured evidence (document IDs, short excerpts, and optionally the matched guideline clause IDs). Dataset statistics are reported in Appendix A; across scenarios, cases contain a median of 23–29 documents, with a long tail up to 106 documents and 1,027,895 characters after OCR/normalization (both observed in CI). We split cases into train/dev/test at the case (bundle) level using stratified sampling by scenario with a fixed random seed to avoid leakage; train/dev are used for offline tree induction and model selection, and all results are reported on the held-out test set. We also maintain a small regression suite for validating post-deployment patches (Section 4.4). Scenario logic is defined by public official guideline documents; for transparency and reproducibility we provide the guideline texts (with clause-level structure; English translations for exposition) in Appendix C. The authoritative sources used in deployment are the original Chinese documents.

We instantiate the same workflow across four backbone LLMs (Qwen3-32B, Qwen3-235B-A22B-Instruct, Qwen3-235B-A22B-Thinking(Yang et al., 2025), and DeepSeek-R1-671B(Guo et al., 2025)) with identical decoding and budget settings unless noted. GuideTree produces short per-document summaries only for routing (Section 3.2); verification primitives always read full document text, and the induced tree executes deterministically with a fixed evaluation order (parallelizable across independent leaves). We compare against long-context prompting (LC), retrieval-augmented generation (RAG), agentic pipelines (ReAct, Plan-and-Execute, CodeAct), multi-agent methods (CoA, ToA, TreeReview), and an expert-authored deterministic workflow

Table 2: Efficiency and stability on the chronic disease scenario with Qwen3-235B-A22B-Instruct. We report per-case Avg/P95 input+output characters (K) and latency, number of LLM calls, and decision consistency over K=5 reruns (temperature = 0.1).

Method	Avg I/O Chars (K) ↓	P95 I/O Chars (K) ↓	Avg Latency (s) ↓	P95 Latency (s) ↓	#LLM Calls ↓	Consistency (%) ↑
LC	420	680	128	205	**1**	78
RAG	160	260	62	105	2	79
ReAct	232	410	217	360	18	90
Plan-and-Execute	140	240	163	275	15	89
CodeAct	353	560	71	125	21	72
CoA	210	360	146	245	8	75
ToA	195	330	154	255	15	76
TreeReview	91	145	168	270	16	73
ExpertTree	82	**110**	**21**	36	9	**99**
GuideTree	**74**	120	22	**34**	9	**99**

(ExpertTree). We report decision quality using Macro-F1 (Table 1); for deployment behavior we measure per-case I/O volume (Avg/P95 input+output characters), number of LLM calls, and consistency (final decision agreement over K=5 reruns with temperature = 0.1). Lower is better for cost and calls; higher is better for Macro-F1 and consistency. For tree induction, (λ, μ) in Eq. 1 are tuned on the dev set per scenario once and kept fixed for all reported results (Appendix B). All prompt templates are fixed across backbones and are reported in Appendix D.

4.2 Results

Table 1 reports Macro-F1 across four review scenarios and four backbone LLMs. Two patterns are consistent. First, ExpertTree is the strongest reference across settings, and GuideTree is second-best in all 16 backbone–scenario combinations, indicating that the induced tree preserves most of the benefit of an expert-authored workflow. Second, the gap to ExpertTree is small and stable: GuideTree reaches 84.5–92.8 Macro-F1 and stays within 0.2–0.6 points (avg. 0.38) across both smaller instruction-tuned models (Qwen3-32B) and large reasoning-oriented backbones (Qwen3-235B-A22B-Thinking and DeepSeek-R1-671B). This robustness suggests that compiling guidelines into a fixed set of evidence-grounded checks reduces sensitivity to backbone-specific prompting behavior and run-dependent decision paths in agentic pipelines. Compared with the strongest non-expert baseline per setting, GuideTree improves Macro-F1 by 3.3–7.6 points. RAG improves over long-context prompting by trimming irrelevant context but can still miss dispersed evidence in heterogeneous records. Agentic pipelines (ReAct and Plan-and-Execute) perform strongly by iterative decomposition and evidence search, yet their adaptive choices can drift across runs and often incur higher operational cost. In contrast, GuideTree achieves comparable decision quality without sequential exploration: the same tree structure and evaluation order are reused, and each leaf returns structured evidence, improving reproducibility and auditability.

Table 2 focuses on the chronic disease scenario with Qwen3-235B-A22B-Instruct under identical decoding and budget settings. I/O volume is approximated by the total number of input+output characters (K) in prompts and outputs (predominantly Chinese text), applied consistently across methods. Under this metric, GuideTree reduces average I/O volume (74K chars) and latency (22s / 34s) while keeping 9 LLM calls, matching ExpertTree and far fewer than iterative agent loops. The gains come from (i) summary-based routing that narrows each primitive to minimal document types and a small candidate pool before full-text verification, and (ii) a deterministic order that prioritizes cheap, high-stop-probability checks to avoid unnecessary downstream work. GuideTree also achieves 99% decision agreement over K=5 reruns, on par with ExpertTree, while several agentic and multi-agent baselines are less consistent. This matters in audit settings: under fixed budgets and long-tail evidence, small changes in ordering or exploratory retrieval can alter evidence consumption and cause decision drift. Fixing the workflow offline and constraining evidence access via minimal interfaces yields a more predictable cost profile and more reproducible evidence paths. Appendix E further reports the induced tree topology (depth/width/leaves), skeleton visualizations, and execution profiles (visited leaves and stop-depth) to quantify realized early stopping.

Table 3: Ablation on chronic disease with Qwen3-235B-A22B-Instruct (temperature $= 0.1$, K=5 reruns).

Variant	F1 (%) ↑	Avg I/O Chars (K) ↓	P95 I/O Chars (K) ↓	Avg Latency (s) ↓	P95 Latency (s) ↓	Consistency (%) ↑
GuideTree (Full)	92.0	74	120	22	34	99
w/o document-summary routing	89.3	155	260	46	78	97
w/o minimal interface $(\mathcal{D}_p, \mathcal{K}_p)$	88.7	190	330	58	102	96
w/o cost-aware split-and-prune	89.8	108	190	31	52	98
w/o deterministic order	89.6	82	140	25	41	88

Table 4: Post-deployment continuous optimization (chronic disease; Qwen3-235B-A22B-Instruct, temperature $= 0.1$).

Error type	#Fail	Fix	Auto	ΔF1
Routing / interface miss	18	67	78	+0.9
Primitive limitation	9	56	40	+0.6
Tree / ordering issue	4	25	0	+0.1
Terminology / mapping	6	60	55	+0.4
Policy ambiguity (human)	5	0	0	+0.0
Overall	**42**	**52**	**61**	**+1.5**

4.3 Ablation study

We ablate each core component by removing it from the same fixed chronic-disease workflow while keeping prompts, budgets, and evaluation identical (temperature $= 0.1$, K=5 reruns), so the deltas reflect deployment-relevant impact.

These deltas isolate how each design choice affects accuracy, cost, and determinism under the same deployment tree. Table 3 shows each component's impact on chronic disease review. Removing document-summary routing increases evidence scanning and drops F1 by 2.7 points while sharply increasing cost. Dropping the minimal interface $(\mathcal{D}_p, \mathcal{K}_p)$ causes the largest degradation (-3.3 F1) and the heaviest Avg/P95 cost because more irrelevant documents are pulled into full-text verification. Without cost-aware split-and-prune, both quality (-2.2 F1) and efficiency worsen, indicating that offline induction is important for a good early-stopping structure under deployment budgets. Finally, removing deterministic order mainly hurts stability: cost changes modestly, but decision agreement drops by 11 points due to order-dependent early stopping and evidence exposure.

4.4 Post-deployment continuous optimization

Table 4 summarizes post-deployment optimization on chronic disease review: #Fail is the number of attributed failures, Fix is the share resolved after patching and re-evaluation, Auto is the share of patches auto-accepted after passing a fixed regression suite, and ΔF1 is the regression-suite gain (v1 vs. v0). The dominant failure mode is routing/interface miss, and it is also the safest to fix: expanding $(\mathcal{D}_p, \mathcal{K}_p)$ often recovers missed evidence and thus yields the highest auto-accept rate. Primitive limitations and terminology/mapping issues are partially fixable but more likely to have broader behavioral impact, reducing auto-accept. Tree/ordering edits are rarely auto-applied due to potential shifts in both semantics and cost, while policy ambiguity is handled via human clarification.

GuideTree uses a gated update lifecycle: for each failure batch, we trace executed paths and evidence to localize faults, propose localized patches (interface, primitive, or small workflow edits), and accept only low-risk changes automatically after regression tests under cost/accuracy tolerances; other changes are reviewed and released as versioned updates with change logs for reproducible evolution.

5 Conclusion

We presented **GuideTree**, a guideline-induced framework that compiles textual policies into executable and auditable review trees for long, heterogeneous medical records. GuideTree maps guideline clauses into primitive conditions, binds them to reusable verification primitives with explicit parameters, and grounds each check via minimal routing interfaces: lightweight document summaries are used only for routing, while final verification always reads full document text and returns structured evidence. To support stable and efficient deployment, GuideTree induces a *fixed* tree offline with a cost-aware split-and-prune procedure that trades off decision utility, execution cost, and structural complexity, enabling early stopping and parallelizable leaf checks. On 1,000 cases across four industrial scenarios and four backbones, GuideTree achieves 84.5–92.8 Macro-F1, improving over the strongest non-expert baselines by 3.3–7.6 points and staying within 0.2–0.6 points of ExpertTree (avg. 0.38). In the chronic disease setting, GuideTree reduces average I/O volume from 420K to 74K input+output charac-

ters (-82% vs. long-context prompting) and average latency from 128s to 22s (-83%), while maintaining 99% decision consistency over $K{=}5$ reruns (and reducing P95 latency from 205s to 34s, -83%). Beyond one-shot evaluation, GuideTree supports controlled post-deployment evolution through failure localization and regression-tested, versioned updates.

Limitations

GuideTree depends on the completeness and quality of input records: missing documents, OCR errors, and inconsistent documentation can cause routing misses or incorrect primitive outcomes. We do not explicitly model layout or non-text evidence (e.g., tables, images, scanned forms) beyond OCR text, which may reduce robustness when key signals are format-dependent. Our approach also assumes policies can be decomposed into checkable conditions with clear evidence requirements; ambiguous or judgment-heavy criteria may still require human escalation. Finally, we have not validated transfer beyond the medical/insurance settings studied here (e.g., legal or contract review), and we do not explore reinforcement learning for optimizing tree structure, evaluation order, or routing policies.

Ethical Statement

Our work involves sensitive medical record data. All records used in this study are de-identified prior to processing, with direct personal identifiers removed (e.g., names, IDs, phone numbers, addresses) under appropriate institutional approval and/or data-use agreements. We follow a data-minimization principle: GuideTree uses short document summaries for routing, and accesses full document text only for the specific checks that require evidence. Data access is restricted to authorized personnel and protected by standard security controls (e.g., access logging and encrypted storage/transfer in our production environment). To reduce privacy leakage in outputs, the system returns only the minimum necessary evidence fields and limited excerpts rather than entire documents. Due to privacy and contractual constraints, the underlying medical records are not publicly released.

Acknowledgments

This work was supported by the National Key R&D Program of China (Grant No. 2023YFF1204102) and the Beijing Nova Program (Grant No. 20250484899).

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, and et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Yuan Chang, Ziyue Li, Hengyuan Zhang, Yuanbo Kong, Yanru Wu, Hayden Kwok-Hay So, Zhijiang Guo, Liya Zhu, and Ngai Wong. 2025. TreeReview: A dynamic tree of questions framework for deep and efficient LLM-based scientific peer review. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 15651–15682, Suzhou, China. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 2978–2988.

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

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume*, pages 874–880.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Song Yu, Xiaofei Xu, Ke Deng, Li Li, and Lin Tian. 2025. Tree of agents: Improving long-context capabilities of large language models through multi-perspective reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 4574–4592, Suzhou, China. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. 2024. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237.

Table 5: Dataset statistics by scenario. #Docs and chars report **Median / Avg / Max** per case. Chars is the total character length (after OCR/text normalization) across all documents in a case.

Scenario	#Cases	#Docs per case			Chars per case		
		Med.	Avg.	Max.	Med.	Avg.	Max.
CI (Critical Illness)	600	29	31	106	181,173	267,961	1,027,895
CD (Chronic Disease)	200	23	28	75	22,206	24,368	68,402
Pre (Pre-existing)	100	25	28	80	151,951	217,184	980,123
Dx (Diagnosis Review)	100	26	27	99	137,612	223,901	890,625

A Dataset statistics by scenario

Table 5 summarizes dataset scale and input heterogeneity by scenario. Each case is a bundle of heterogeneous documents (e.g., notes, reports, prescriptions, and claim materials). We report the **Median / Avg / Max** number of documents and total characters per case (after OCR/text normalization), highlighting substantial cross-case variation and long-tail inputs. We place these statistics in the appendix to keep the main text focused on methodology and evaluation outcomes.

B Utility weights for cost-aware split-and-prune

We tune the utility weights (λ, μ) in Eq. 1 on the dev set per scenario with a small grid, selecting the pair that maximizes dev Macro-F1 while keeping expected cost and tree size within deployment tolerances. The chosen weights are then fixed for all test runs and all backbones within the scenario. Table 6 reports the selected (λ, μ) for each scenario. Table 7 further provides a sensitivity sweep on chronic disease to illustrate the cost–structure–quality trade-off.

Overall, GuideTree is robust to moderate changes of (λ, μ) (Table 7): Macro-F1 varies within 0.6 points, while λ and μ provide smooth control of cost and structural complexity. Smaller λ (weaker cost penalty) tends to yield deeper trees

Table 6: Chosen utility weights for Eq. 1 (tuned on dev; fixed on test).

Scenario	λ	μ
CI	0.90	0.06
CD	0.80	0.05
Pre	1.00	0.05
Dx	0.70	0.04

Table 7: Sensitivity of (λ, μ) on chronic disease (dev; Qwen3-235B-A22B-Instruct). Cost is approximated by Avg/P95 I/O chars (K) as in Table 2.

(λ, μ)	F1	Avg	P95	Avg Lat.	Depth	#Leaves
(0.40, 0.05)	92.2	79	132	24	9	28
(0.80, 0.025)	92.1	76	125	23	9	27
(0.80, 0.05)	92.0	74	120	22	8	24
(0.80, 0.10)	91.7	71	115	21	6	18
(1.60, 0.05)	91.6	68	110	20	6	17

with slightly higher F1 but higher cost; larger λ produces shallower trees with lower cost at a small F1 drop. Increasing μ reduces depth/leaves and tail cost.

C Guideline Texts (Public Official Documents)

This appendix provides the guideline texts used to define scenario logic with clause-level structure for traceability. English translations are provided for exposition and research use only; the authoritative sources are the original Chinese official documents used in deployment.

C.1 Chronic Disease Review Guidelines

Hypertension Guideline

```
This text is translated from a Chinese
    national guideline for outpatient
    chronic disease eligibility (
    hypertension).
It is for research/demo use only. The
    authoritative source is the original
    Chinese guideline.

---

Eligibility requires BOTH:

I. Confirmed diagnosis of essential (primary)
     hypertension at a secondary or higher-
    level hospital
(ICD I10). Secondary hypertension (I15) is
    excluded. Diagnosis name or code may be
    used.

II. BOTH of the following:
1) Diagnostic criteria met (per latest
    national guideline; home BP criteria may
     be excluded). Either:
    (1) Office BP >= 140/90 mmHg on three non
    -consecutive days; OR 24h ambulatory BP
    >= 130/80 mmHg
        (daytime >= 135/85, nighttime >=
    120/70); evidence may come from
    admission/discharge notes,
        temperature sheets, or ambulatory BP
    reports.
    (2) If history documents hypertension
    plus antihypertensive use, provide three
     non-consecutive BP
        readings (e.g., temperature sheet or
    ambulatory BP); diagnosis may be
    accepted even if readings
        do not exceed the usual cutoff.
2) At least ONE target-organ damage OR
    clinical disease, supported by evidence:
    (1) Target-organ damage (test/imaging
    required; not progress notes only):
        ů LVH: ECG Sokolow-Lyon >3.8 mV or
    Cornell product >244 mVůms, or echo LVH/
    LVMI >=109 g/mš (men)
           or >= 105 g/mš (women).
        ů Carotid/vascular ultrasound: IMT
    >=0.9 mm or plaque.
        ů cfPWV >=10 m/s or baPWV >=18 m/s.
        ů ABI <0.9.
        ů eGFR 30-59 mL/(minů1.73 mš) or mild
     creatinine elevation (men 115-133 umol/
    L; women 107-124).
        ů Microalbuminuria: ACR 30-300 mg/g
    or AER 30-300 mg/24h.
    (2) Clinical disease (any one):
        ů Cerebrovascular disease:
    intracerebral hemorrhage, ischemic
    stroke, TIA; dx or ICD I63-I64.
        ů Cardiac disease: prior MI (I21-I23)
    , angina (I20), coronary
    revascularization, chronic HF
           (I11.0/I13.0/I13.2/I50 excluding
    acute failure), or atrial fibrillation.
        ů Kidney disease: diabetic kidney
    disease or renal impairment (N17-N19)
    plus eGFR < 30, or
           creatinine (men >=133, women >=124
    umol/L), or 24h protein >=300 mg/24h (
    two results required).
        ů Peripheral artery disease: dx or
    records indicating atherosclerosis/
    plaque/stenosis/occlusion/ischemia.
        ů Retinopathy: retinal hemorrhage/
    exudate/papilledema (excluding trauma or
     diabetic retinopathy).
        ů Diabetes: diagnosis of diabetes.
```

Diabetes Guideline

```
This text is translated from the relevant
    Chinese national guideline for
    outpatient chronic disease eligibility (
    diabetes). It is for demonstration and
    research use only. The authoritative
    source is the original Chinese guideline.


---

Eligibility is satisfied if either [Type 1
    diabetes] or [Type 2 diabetes] criteria
    are met; the two are reviewed separately.


Type 1 diabetes criteria:
(1) Confirmed diagnosis of type 1 diabetes at
     a secondary or higher-level hospital (
    ICD category E10).
(2) All of the following:
ů Medical records include typical diabetes
    symptoms (e.g. polyuria, polydipsia,
    polyphagia, weight loss) or a history of
     acute ketoacidosis or hyperglycemic
    hyperosmolar state requiring
    hospitalization.
ů Random plasma glucose >= 11.1 mmol/L, or
    fasting plasma glucose >= 7.0 mmol/L on
    three or more occasions (capillary
    glucose excluded from criteria), or OGTT
     2h >= 11.1 mmol/L, or HbA1c >= 6.5%.
ů History of at least six months (see
    admission/prior history). Records must
    support at least two of: age under 30;
    low or absent fasting or postprandial
    serum insulin or C-peptide; positive
    autoimmune markers (e.g. GAD, ICA, IA-2A
    ).

Type 2 diabetes criteria:
(1) Confirmed diagnosis of type 2 diabetes at
     a secondary or higher-level hospital (
    ICD category E11).
(2) All of the following:
ů Random plasma glucose >= 11.1 mmol/L, or
    fasting plasma glucose >= 7.0 mmol/L on
    three or more occasions, or OGTT 2h >=
    11.1 mmol/L, or HbA1c >= 6.5%.
ů History of at least one year (see admission
    /prior history).
ů At least one complication from: diabetic
    kidney disease, diabetic retinopathy,
    diabetic neuropathy, diabetic lower-
    extremity arterial disease, diabetic
    foot (each with specified diagnostic and
     test criteria in the full guideline).
```

C.2 Diagnosis Review Guidelines

Diagnosis Review Guideline

```
# English version of ICD classification
    concepts and conventions

This text is translated from the relevant
    Chinese national guideline on ICD-10
    classification and coding. It is for
    demonstration and research use only. The
     authoritative source is the original
    Chinese guideline.

------------

[Concepts of the given disease classification
    ]
Classification axis: The classification axis
    is the disease feature used for
    classification. In ICD, four main types
    are used: etiology, anatomical site,
    clinical manifestation (symptoms, signs,
     stage, type, sex, age, acute/chronic,
    onset, etc.), and pathology. ICD is thus
     multi-axis. Usually only one axis is
    used per level, but two axes may appear
    at one level in some chapters.

[Chapters with strong priority]
Chapter 15 - Pregnancy, childbirth and the
    puerperium;
Chapter 16 - Certain conditions originating
    in the perinatal period.

[Chapters for last-resort classification]
Chapter 18 - Symptoms, signs and abnormal
    clinical and laboratory findings, not
    elsewhere classified;
Chapter 21 - Factors influencing health
    status and contact with health services.
For conditions in these chapters, when a
    clear etiology or other condition exists,
     their codes are used only as additional
     codes.

[Additional coding chapter]
Chapter 20 - External causes of morbidity and
     mortality.
In ICD-10 this chapter is in the main
    classification; in statistics its codes
    are usually excluded to avoid double-
    counting injury/poisoning with Chapter
    19.

[Terms, symbols and abbreviations in ICD-10]
(1) Terms: category (3-digit), subcategory
    (4-digit), subcategory with 5th digit,
    residual categories ("other"/"
    unspecified"), dual classification (
    dagger/asterisk), main and additional
    code, combined code.
(2) Symbols: square brackets [synonyms, notes
    ], round parentheses (optional modifiers
    ), braces, colon, asterisk/dagger, # and
     * in index tables.
(3) Abbreviations: NOS (not otherwise
    specified), NEC (not elsewhere
    classified).

[Lookup method]
1. Determine the lead term.
2. Find the code in Volume 3 index under the
    lead term and modifiers.
3. Verify the code in Volume 1 category table
    .

[Lead term selection]
```

```
Lead terms are usually the main clinical
    manifestation (often at the end of the
    diagnosis), or etiology, or eponym/place
     name. Specific rules apply for "disease
    ", anatomy, pregnancy/delivery/
    puerperium, tumors, and injury types.

------------

In the diagnosis review scenario, tasks
    instantiate these ICD-10 concepts and
    conventions for specific target codes.
```

D Prompts

This appendix lists all prompts used in the Guide-Tree pipeline. In our implementation, prompt templates are written in Chinese; we translate them into English here for readability and include the English versions below.

D.1 Document Summarization

Document Summarization

```
Please summarize the following medical
    document in one sentence, strictly
    within {max_chars} characters. Output
    only the summary text-no title, no
    explanation, and do not end with
    punctuation.
Document type: {doc_type}

Document content:
{ocr_text}

Summary:
```

D.2 LLM Verification (Primitive)

LLM Verification

```
You are a medical document audit expert.
    Based on the audit criteria below,
    determine whether the document content
    satisfies the condition.

Audit point description:
{audit_desc if audit_desc else audit_content}

Detailed requirements:
{audit_content}

Document content to be audited:
{combined_text}

Please make a strict judgment: does the
    document content satisfy the audit point
     requirements above?

Return in JSON format:
{
  "satisfied": true/false,
  "evidence": ["evidence snippet 1", "
    evidence snippet 2", ...],
  "reasoning": "Reason for the decision (one
    sentence)"
}

Notes:
ů satisfied: whether the audit requirement is
     met (true/false)
ů evidence: a list of key evidence snippets
    supporting the decision (extracted from
    the document)
ů reasoning: briefly explain why you made the
     decision
ů Return JSON only, with no additional
    explanation
```

D.3 Tree Initialization

Tree Initialization

```
You are a medical audit rule analysis expert.
     Based on the guideline text below,
    generate an audit tree structure. The
    tree will be used later to perform yes/
    no verification on medical record
    documents and aggregate the results.

Guideline text (full original text; when
    generating nodes you must always adhere
    to this text-do not simplify or omit any
     key determination statements):
{guideline_text}

**Hard constraints (must follow)**:
1. **Total number of nodes must not exceed {
    max_nodes}** (including both logic and
    audit nodes).
2. **Tree depth must not exceed {max_depth}
    levels** (the root is level 1; its
    children are level 2; no deeper levels).
3. **Total number of leaf nodes (audit nodes)
     must not exceed {max_leaves}**.
4. **The root node must be of type "logic"**,
     representing the aggregation of the
    overall determination logic.

**Do not lose key information** (while
    satisfying the constraints above):
ů The guideline's main determination paths
    and each complication type must be
    reflected in the tree. Multiple sub-
    conditions under the same type may be
    merged into a single audit node, whose
    content can summarize the type-level
    requirement (e.g., "Whether the
    diagnostic and lab criteria for diabetic
     nephropathy are met"). It will be
    expanded later during refinement.
ů For example, if complications include (1)
    nephropathy (2) retinopathy (3)
    neuropathy, you may create one audit
    node per complication type, or place
    multiple audit nodes under a logic(OR).
    Do NOT create a separate node for every
    sub-condition, to avoid exceeding node
    count or depth limits.

**Keep the tree as simple as possible**:
```

```
ů Clear hierarchy, explicit AND/OR, root as a
     logic node. Within node/depth limits,
    prioritize expressing "what AND/OR what"
     and avoid overly deep or overly fine-
    grained trees.

Node types:
1. **"audit" (audit node)**: a specific audit
     check that MUST be a **binary yes/no**
    proposition
    ů desc and content must be phrased as a
    proposition whose truth can be judged,
    such as "Whether ..." / "... whether
    meets ..."
    ů **Forbidden**: open-ended task
    descriptions such as "extract...", "
    obtain...", "find..." etc.
    ů Correct example: "Whether the admission
     note states the full official name of
    the hospital issuing the diagnosis"
     Incorrect example: "Extract the full
    official name of the hospital issuing
    the diagnosis from the admission note"
    ů This is required so logic nodes (AND/OR
    ) can aggregate based on true/false
    results
2. **"logic" (logical aggregation node)**:
    combines multiple audit nodes via
    logical relationships (AND/OR)
    ů When type="logic", you MUST specify the
     field logic_type with value "AND" or "
    OR"
    ů "AND" means all child nodes must be
    satisfied (all true)
    ů "OR" means at least one child node must
     be satisfied (at least one true)

**Important: node desc/content must not
    simplify or omit key guideline
    information**: for a node corresponding
    to a guideline item, its content must
    retain the item's complete determination
     statements (e.g., diagnostic criteria
    for a complication, lab requirements,
    codes, etc.). Do not generalize,
    abbreviate, or omit, so that subsequent
    auditing is fully grounded.

Edges represent parent-child relationships:
    each edge uses from=child node id, to=
    parent node id. The root node appears
    only in "to", never in "from".

Return ONLY the following JSON (no other
    explanation):
{
  "nodes": [
    {
      "id": "root",
      "type": "logic",
      "logic_type": "OR",
      "desc": "Brief description of the
    overall determination logic",
      "content": "e.g., satisfies either Type
     1 or Type 2 diabetes determination
    criteria"
    },
    {
      "id": "audit_1",
      "type": "audit",
      "desc": "Whether-type description (one
    sentence, decidable yes/no)",
      "content": "Specific audit requirement
    (must be a decidable true/false
    proposition; may preserve the guideline'
    s original wording with complete
    determination statements)"
    }
  ],
  "edges": [
    { "from": "audit_1", "to": "root", "
    weight": 1 }
  ]
}

Notes:
ů Return JSON only; no extra explanation
ů **Hard limits: total nodes <= {max_nodes},
    depth <= {max_depth}, leaf nodes <= {
    max_leaves}**; if the guideline is long,
     merge similar conditions (e.g., one
    audit node per complication type); it
    will be expanded later
ů **Important: the root node must be of type
    "logic"**
ů **Important: each audit node's desc and
    content must be a binary yes/no
    proposition**, phrased as "Whether..." /
     "... whether meets ..."; do NOT use "
    extract/get/find" style open-ended
    descriptions
ů **Important: node desc/content must retain
    the complete corresponding determination
     statements from the guideline**,
    without self-made generalization,
    abbreviation, or omission
```

D.4 Audit Node Split (Induction)

Audit Node Split

```
You are a medical audit rule analysis expert.
     You need to split a composite audit
    condition into multiple atomic audit
    conditions.

{guideline_section}

Audit node to be split:
ů Description: {desc}
ů Detailed content: {content}
ů Current document types: {doc_types if
    doc_types else "Not specified"}

Please analyze this audit condition and split
     it into multiple independent sub-
    conditions. Each sub-condition should be
     the smallest unit that can be judged
    independently.

**Important constraints**:
1. **Each sub-condition's desc and content
    must be a binary yes/no proposition**,
    phrased as "Whether...", "... whether
    meets ...", "... whether exists ...",
    etc. Open-ended task descriptions such
    as "extract...", "obtain...", "find..."
```

```
    are forbidden.
2. **The sub-condition content must not
    simplify or omit key guideline
    information**: for any sub-condition
    corresponding to a guideline item, its
    content must retain the item's complete
    determination statements (e.g.,
    diagnostic criteria for complications,
    lab requirements, codes, etc.),
    consistent with or fully quoting the
    Guideline original text above. Do not
    generalize, abbreviate, or omit.
3. The number of sub-conditions should be
    reasonable (typically 2-4). Do not over-
    split.

Return in JSON format:
{
  "logic_type": "AND" or "OR",
  "sub_conditions": [
    {
      "desc": "Brief description of sub-
    condition 1 (yes/no proposition)",
      "content": "Detailed content of sub-
    condition 1 (must be a decidable true/
    false proposition; retain the complete
    guideline determination statements)",
      "doc_types": ["Document type 1", "
    Document type 2"]
    },
    {
      "desc": "Brief description of sub-
    condition 2 (yes/no proposition)",
      "content": "Detailed content of sub-
    condition 2 (must be a decidable true/
    false proposition; retain the complete
    guideline determination statements)",
      "doc_types": ["Document type 1"]
    }
  ]
}

Notes:
ů logic_type: use "AND" if sub-conditions are
     in an "and/also/simultaneously"
    relationship; use "OR" if they are in an
     "or/either" relationship
ů sub_conditions: list of sub-conditions;
    each sub-condition should be
    independently executable for auditing
ů doc_types: required document types for each
     sub-condition (infer from the original
    doc_types or keep as-is)
ů **Important: each sub-condition's desc/
    content must be a binary yes/no
    proposition**; do NOT use "extract/get/
    find" style open-ended descriptions
ů **Important: sub-condition content must
    retain the complete corresponding
    determination statements from the
    guideline**, without self-made
    generalization or omission
ů Return JSON only, with no additional
    explanation
```

D.5 Leaf Binding (Routing Interface)

Leaf Binding

```
You are a medical audit rule analysis expert.
     Please analyze the audit node
    description below and select the
    required document types from the
    provided list.

Audit node description:
ů desc: {audit_desc}
ů content: {audit_content}

Available document types (you MUST select
    from the list below using the full names
    ):
{doc_types_list}

Return JSON:
{
  "doc_types": ["Document type 1", "Document
    type 2"],
  "summary_keywords": ["keyword 1", "keyword
    2"]
}

Notes:
ů doc_types MUST be selected from the
    available document type list above,
    using the full names; do not use
    abbreviated names
ů summary_keywords is optional and used for
    quick filtering in document summaries
    (2-5 keywords). If none, return an empty
     array []
ů Return JSON only, with no additional
    explanation
```

E GuideTree Structure and Execution Profiles

This appendix provides a deeper view of **what the induced workflows look like** (tree topology and size) and **how they execute in practice** (visited leaves, reached depth, and early stopping). The main paper reports end-to-end review quality and system-level efficiency. Here we add structural and execution-level evidence to address two deployment-facing questions:

- **Complexity control.** Do induced trees remain compact and inspectable as guideline logic grows, or do they become too deep/wide to audit and maintain?
- **Realized savings.** With a fixed tree, do we actually evaluate only a subset of leaves for most cases via short-circuit evaluation and cheap-first ordering, especially under long-tail records?

We report topology statistics (Table 8), a qual-

itative skeleton visualization (Figure 2), and execution profiles that quantify early stopping (Table 9 and Table 10). We keep the chronic disease (CD) tree depth and leaf count consistent with Appendix B (Depth=8, #Leaves=24) to align structure reporting with the utility-weight sensitivity study.

E.1 Tree size and topology

Table 8 summarizes the induced GuideTree topology by scenario. We report total nodes (logic + audit leaves), number of logic nodes, number of audit leaves, maximum depth, and maximum width (the largest number of nodes at any depth level). These metrics jointly characterize **auditability** (how easily reviewers can trace conditions), **maintainability** (whether patches remain local), and **parallelizability** (how many independent checks can be evaluated concurrently).

Scenario-dependent structural complexity. Total nodes range from 38 (Dx) to 60 (CI), reflecting differences in guideline logic. CI typically contains more heterogeneous eligibility pathways and exception handling, while Dx focuses on a smaller set of coding conventions and therefore yields a more compact tree. CD and Pre sit in the middle: they require non-trivial evidence aggregation, but admit relatively modular decompositions (e.g., diagnosis confirmation, measurement/time-window checks, and complication/evidence branches).

Depth as a proxy for sequential dependency. Depth ranges from 7 to 9. Depth matters for interpretability (nested conditions) and update risk (deep nesting can amplify the impact of structural edits). Importantly, depth does not directly translate to runtime cost because most cases terminate early via short-circuit evaluation; we therefore complement depth with execution profiles in Section E.3.

Width and concurrency. Maximum width ranges from 9 (Dx) to 14 (CI). Width serves as a practical proxy for available parallelism: nodes at the same depth often correspond to independent primitives under the same AND/OR parent. In deployment, these checks can be issued concurrently (subject to system limits), making latency closer to the slowest visited leaf rather than the sum of all leaves.

Table 8: GuideTree structural statistics by scenario.

Scenario	#Nodes	#Logic	#Leaves	Depth	Max width	Leaf ratio
CI	60	28	32	9	14	0.53
CD	44	20	24	8	10	0.55
Pre	50	24	26	8	11	0.52
Dx	38	18	20	7	9	0.53

Leaf ratio and decomposition granularity. The leaf ratio is stable across scenarios (roughly 0.52–0.55), suggesting a consistent decomposition pattern: about half of the nodes are leaf primitives (direct evidence checks) and half are logic nodes that structure stopping behavior and traceable aggregation. This balance is desirable for maintenance: most post-deployment patches are leaf- local (e.g., routing interface edits, terminology mapping), while logic scaffolding stays stable.

Notes. Depth is measured with the root as depth 1. Max width is the maximum number of nodes at the same depth level. These are structural properties of the induced workflow and are independent of backbone choice: the same fixed tree is executed for all backbones.

E.2 Tree visualization

We provide a qualitative visualization of the induced GuideTree for the chronic disease scenario, focusing on the **diabetes eligibility** workflow. To keep the figure readable, we visualize only the skeleton (logic structure and leaf short names). The full leaf contents (including clause bindings, primitive parameters, and routing interfaces) are provided via the released configuration files.

Induced diabetes GuideTree skeleton (Figure 2). The induced tree follows the guideline structure: it first verifies diagnosis confirmation (e.g., ICD E10/E11 and hospital level), then checks glycemic criteria (FPG/OGTT/HbA1c) and required history duration, and finally routes to complication evidence branches when needed. This modular design supports early stopping for obvious negatives and limits full-text reads to a small set of targeted documents.

E.3 Execution profile and early stopping

GuideTree is designed to exploit (i) **short-circuit evaluation** under $\wedge/\vee$ and (ii) **cheap-first deterministic ordering** that prioritizes low-cost, high-stop-probability checks. This section quantifies how these design choices translate into realized execution savings on the test set.

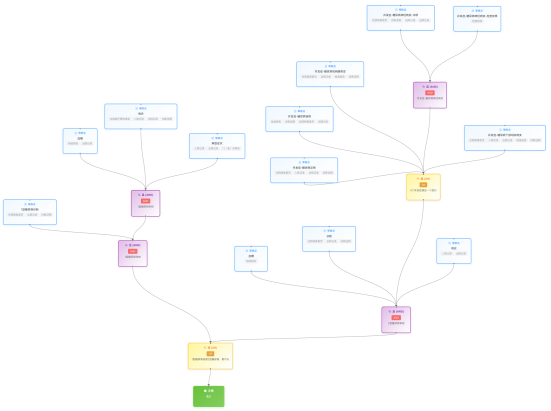

Figure 2: Induced GuideTree skeleton for diabetes eligibility review (chronic disease scenario).

Table 9: Execution profile on the test set.

Scenario	Avg leaves	P95 leaves	Max	Avg depth	P95 depth	Early-stop %
CI	12.4	20	29	6.2	8	92
CD	9.1	15	20	5.0	7	96
Pre	10.0	17	22	5.6	7	94
Dx	7.3	12	16	4.8	6	97

Table 10: Cumulative stop-depth distribution on chronic disease.

Depth $\leq$	1	2	3	4	5	6	7	8
% cases	0.0	12.0	28.0	52.0	73.0	86.0	95.0	100.0

Table 9 reports how many leaves are visited per case and how deep evaluation typically proceeds. We also report the early-stop rate: the percentage of cases that terminate without evaluating the full set of reachable leaves. These execution-level metrics complement system-level I/O and latency statistics (Table 2) by directly showing *how much of the workflow is exercised* per case.

Most cases evaluate a minority of leaves. Across scenarios, the average visited leaves (7.3–12.4) are well below the total leaf counts (20–32 in Table 8). Even at P95, visited leaves remain substantially below the full leaf count. This indicates that the induced structure and ordering convert guideline logic into a decision procedure that stops early for most inputs, rather than a checklist that always scans everything.

Reached depth is meaningfully smaller than structural depth. Average reached depth (4.8–6.2) is also below maximum structural depth (7–9). Many cases terminate within upper-level prerequisites, and only hard cases propagate into deeper subtrees. This supports the claim that efficiency gains come not only from routing, but also from inducing a structure that front-loads decisive checks.

Tail cases and ambiguity. Max visited leaves correspond to difficult cases where multiple branches remain plausible and evidence is dispersed. These cases are also more likely to trigger routing/interface misses or policy ambiguity (Table 4). The key deployment result is that the tail is rare (captured by P95) and does not dominate average cost.

Interpretation. High early-stop rates imply that most cases terminate after evaluating only a subset of leaves. This aligns with the low Avg/P95 I/O and latency observed in Table 2 (CD): the workflow's AND/OR structure and cheap-first ordering systematically avoid unnecessary downstream checks.

Stop-depth distribution (chronic disease). To provide a more interpretable view, Table 10 reports the cumulative distribution of stop depth on the chronic disease test set. More than half of cases stop by depth ≤ 4, and most cases stop before entering the deepest evidence-heavy branches, consistent with GuideTree's stable cost profile.

E.4 Minimal routing interface analysis

A key design choice in GuideTree is the **minimal routing interface** for each primitive: $\mathcal{D}_p$ (a small set of document types) and $\mathcal{K}_p$ (summary-level cues) used to select a few candidates before full-text verification. The objective is to achieve both (i) **coverage** (do not miss relevant evidence) and (ii) **cost control** (avoid scanning irrelevant documents), while keeping access patterns predictable for audit.

We report interface sizes and candidate counts in Table 11, and the per-scenario record heterogeneity context (avg. docs/case and avg. chars/case) in Table 12. Together they quantify how aggressively primitives restrict evidence access relative to case-level document diversity.

Interfaces are small relative to record heterogeneity. Avg $|\mathcal{D}_p|$ ranges from 1.9 to 2.6, while cases contain $\sim$27–31 documents on average (Appendix A). Thus, each primitive typically restricts itself to only 2–3 document types out of many. Dx has the smallest interfaces because coding evidence concentrates in a limited set of notes; CI and Pre have slightly larger interfaces because policy evidence spans more heterogeneous sources (e.g.,

Table 11: Routing interface sizes and candidate selection (aggregated across primitives).

Scenario	Avg $\lvert\mathcal{D}_p\rvert$	P95 $\lvert\mathcal{D}_p\rvert$	Avg cand./leaf	P95 cand./leaf
CI	2.6	5.0	3.4	8.0
CD	2.1	4.0	2.8	6.0
Pre	2.4	4.0	3.1	7.0
Dx	1.9	3.0	2.5	5.0

Table 12: Record heterogeneity context for interface minimality.

Scenario	Avg docs/case	Avg chars/case (K)
CI	31	268.0
CD	28	24.4
Pre	28	217.2
Dx	27	223.9

procedures and imaging).

Candidate selection is aggressively selective. Avg candidates per leaf are 2.5–3.4, with P95 below 8. Even after selecting document types, summary cues further narrow the evidence pool before full-text verification, preventing pathological "pull everything" behaviors common in sequential agentic exploration under long, noisy records.

Link to observed failure modes. The dominant post-deployment failure type is routing/interface miss (Table 4), which is an expected risk of aggressive minimality. However, it is also the easiest to patch safely: expanding $\mathcal{D}_p$ or refining $\mathcal{K}_p$ is a localized change with clear attribution and limited blast radius, explaining the high auto-accept rate for such patches.

E.5 Induction trace (chronic disease)

To improve transparency of the offline split-and-prune procedure, Table 13 shows an abridged induction trace for chronic disease: each step applies the best positive-utility split (Eq. 1) and updates local statistics. We report $\Delta\mathcal{I}$ (empirical uncertainty reduction), ΔC (expected cost change), $\Delta\Omega$ (structural penalty), and utility U; we also track the resulting depth and number of leaves.

Early steps deliver the largest utility. The first few splits typically separate prerequisites (e.g., diagnosis confirmation) from evidence-heavy modules (e.g., organ damage vs. clinical disease). These splits yield large $\Delta\mathcal{I}$ and improve early stopping by front-loading decisive checks.

Later steps fine-tune structure under deployment constraints. As the tree grows, marginal gains shrink and penalties become more influential; the search stops when no split remains positive-U. This behavior is consistent with the smooth depth/leaf trade-offs observed in the sensitivity study (Appendix B).

Reordering and minor merges matter for stability. The final step shown emphasizes cheap-first reordering and minor merges. While these edits contribute less to $\Delta\mathcal{I}$, they mainly improve reproducibility by fixing deterministic evaluation order, consistent with the ablation where removing deterministic order mainly hurts consistency (Table 3).

Table 13: Abridged split-and-prune induction trace on chronic disease.

Step	Split applied (node short name)	$\Delta\mathcal{I}$	ΔC	$\Delta\Omega$	U	Depth	#Leaves
1	`Dx_confirm` $\wedge$ `Evidence`	6.8	1.9	0.8	5.25	6	14
2	`BP_criteria (OR branches)`	5.2	1.6	0.7	3.88	7	16
3	`Organ_damage vs Clinical_disease`	4.9	1.5	0.8	3.67	7	18
4	`Kidney_evidence (labs vs dx)`	3.7	1.1	0.6	2.79	8	21
5	`Cardiac_evidence (dx vs procedure)`	3.4	1.0	0.5	2.58	8	23
(Omitted intermediate steps for brevity.)							
14	`Final: minor merge + reorder`	1.2	0.4	0.3	0.88	8	24

E.6 Clause-level traceability examples

Beyond accuracy and cost, medical and insurance review systems require **traceability**: reviewers should be able to map each decision to the relevant policy clause(s) and to the concrete evidence used. GuideTree supports this by attaching clause identifiers to primitives during compilation and propagating them to evidence items returned at inference time.

Why clause-level traceability matters. Clause-to-evidence links serve three practical purposes: (i) **auditing and compliance** (decisions are grounded in explicit clauses rather than implicit model behavior), (ii) **debugging and maintenance** (localize failures to a clause/primitive pair), and (iii) **safe updates** (regression tests can be organized around clauses and their bound primitives). In deployment, we store per-version workflow metadata so past decisions can be replayed under the exact same configuration.