# Taming Guidelines in the Wild

Stanford CS224N Custom Project

**Anuj Iravane**
Stanford University SCPD
anuj@anterior.com

## Abstract

Decision-making using pre-defined guidelines is a prevalent task in various domains, including insurance claim handling, clinical diagnosis, and student assignment evaluation. While various solutions utilizing Large Language Models (LLMs) have been developed, these solutions are often domain-specific and tailored to specific guideline formatsLi et al. (2023). Moreover, certain domains, such as medical diagnosis and claim handling, require determinations to provide a 'proof of work' and a high degree of auditability, which most general LLM-based free text-to-free text implementations struggle to achieve. LLMs also often exhibit poor and inconsistent performance in accurately applying complex guidelines.

This paper proposes a novel approach to address these challenges by introducing a unified symbolic representation called GDT, designed to be able to represent all types of guidelines in a succinct format. One of the significant advantages of GDT is its ability to be programmatically executed, enabling automated decision-making processes based on these representations. Additionally, we present a dataset comprising pairs of free-text and GDT representations, alongside a novel technique for synthetically generating such data. Furthermore, we present an aligned model that achieves improved performance on this task as compared to baseline foundational models.

## 1 Key Information

- Mentor: Soumya Chatterjee
- External Collaborators / Sharing Project (if you have any): No

## 2 Introduction

Rule-based decision-making systems are ubiquitous. They are found in software as if/else statements, medical guidelines, take-off checklists, legal proceedings, and more. Decision-making processes across various industry domains frequently rely on structured guidelines to ensure consistency, accuracy, and fairness. Traditionally, these tasks—ranging from insurance claim evaluations to clinical diagnoses—are performed by human experts who interpret and apply complex sets of guidelines to make determinations based on available evidence. In recent years, there has been a significant shift towards automating these tasks using Large Language Models (LLMs).

The typical framework for such automation involves inputting guidelines and evidence into an LLM, which then outputs a set of outcomes.

Despite the promise of LLMs in automating guideline-based decision-making, several critical shortcomings have been identified. Firstly, LLMs generally exhibit poor performance on the complex task of transforming guidelines and evidence into determinations. The performance notably deteriorates as the complexity of the guidelines increases. This degradation is believed to stem from the inherent difficulty of the task, which requires the model to not only answer questions based on the guidelines

but also logically map these answers to reach a final determination. Such tasks present multiple potential failure modes, particularly as the complexity and nuance of the guidelines increase.

Another significant challenge is the training and fine-tuning of LLMs for specific domains. Due to the coupling of two subtasks— question answering and logical deduction—within the decision-making process, it becomes cumbersome to improve performance of one of the subtask independently without impacting the other. This is especially problematic when adapting traditional question-answering tasks to specific domain requirements, where each domain might have unique characteristics and requirements (for e.g different needs for uncertainty modeling in clinical vs legal domains), but the process for logical deduction is largely the same.

Moreover, in domains such as law and medicine, the stakes for accuracy and consistency are exceptionally high. Not only must the determinations made by LLMs achieve a high degree of precision, but they also need to provide accountability. This is often expected in the form of a 'proof of work' that adheres to a structured representation, which can be challenging when dealing with inputs that are initially in a loosely structured free-text format.

Given these challenges, there is a clear need for a more robust, adaptable, and transparent approach to automating guideline-based decision-making. Through this paper, we propose a novel framework that addresses these issues by introducing a unified symbolic representation called GDT (Guideline Decision Tree), which is designed to standardize the representation of guidelines across various domains and enhance the auditability and performance of automated decision-making systems. We also demonstrate how LLM's may be used to convert any and all free-text guidelines into GDT's.

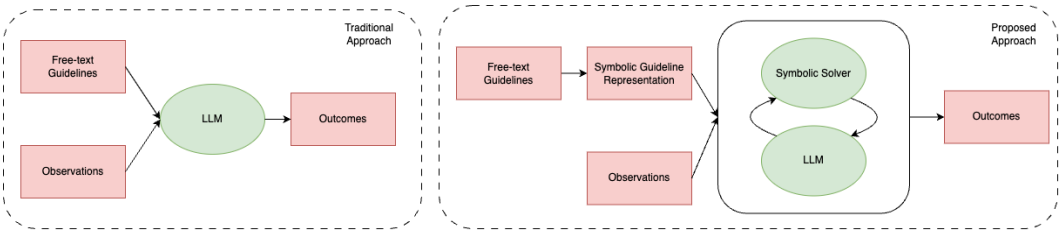


Figure 1: Left: Classical Approach. Right: Proposed Approach

## 3 Related Work

Numerous studies have been performed to evaluate performance of foundational LLM's on logical reasoning tasks. Methods like ToT and CoT prompting have been widely adopted to significantly improve base reasoning capabilities on most tasks. However, in domains requiring reasoning with a minimal margin of error, symbolic systems are still predominantly used for logical reasoning. In recent months, systems like AlphaGeometry have used LLM's in conjunction with symbolic reasoning to solve complex problems. Trinh et al. (2024) The authors of LINC provide promising evidence for how logical reasoning over natural language can be tackled through jointly leveraging LLMs alongside symbolic provers.Olausson et al. (2023)

## 4 Approach

The objective of the Guideline, Evidence $\rightarrow$ Outcomes task is to identify the outcomes from a given guideline based on some evidence from the world at a particular instant in time.

We posit that splitting this task into two distinct subtasks— (1) creating symbolic guideline representations from free text and (2) executing these symbolic representations based on provided evidence— is preferable for several reasons as outlined above. For the purpose of this project, we explore solely the first subtask of creating symbolic guideline representations. The key, original contributions of this project include:

- An evaluation of different forms of symbolic guideline representations.

- A proposed unified symbolic guideline representation, GDT (Guideline Decision Tree), which can encapsulate the logic found in all types of free-text guideline representations.
- A library for synthetically generating free-text and GDT pairs, converting and parsing GDT's into their textual representations, and executing and comparing GDT's.
- Models fine-tuned for the free-text to GDT transformation task.
- Evaluations of both baseline and fine-tuned models.

**Guidelines as Logical Expressions**

A guideline can be interpreted as a set of predicates, a set of outcomes, and a mapping function between them. A predicate is an independently verifiable truth statement about the world that can be either true or false, and each outcome is a potential result of the guideline determination.

The mapping function yields a list of 'achieved' outcomes given the truth values for its predicates. Formally, a guideline can be represented as follows:

- Predicates: $p_1, p_2, \ldots, p_n$
- Outcomes: $o_1, o_2, \ldots, o_m$
- Mapping function: $f : \{0,1\}^n \to \{0,1\}^m$

The mapping function can be seen as $m$ different mapping functions for each of the outcomes:

$$f_i : \{0,1\}^n \to \{0,1\} \text{ for } 1 \leq i \leq m$$

Each mapping function $f_i$ is a logical expression involving $p_1, p_2, \ldots, p_n$.

A common way of representing a logical expression is through its Disjunctive Normal Form (DNF), which consists of disjunctions (OR) of conjunctions (AND) of predicates or their negations.

The Disjunctive Normal Form (DNF) for outcome $o_i$ based on predicates $p_1, p_2, \ldots, p_n$ is:

$$o_i \leftrightarrow \bigvee_{\substack{k \in \{1,\ldots,2^n\} \\ \text{where } o_i \text{ is } 1}} \left( \bigwedge_{j=1}^{n} p_j^{(k)} \right)$$

However, DNF is inefficient for representing guidelines due to the exponential growth of conjunctive statements, resulting in $2^n$ combinations. A more succinct representation is the minimal DNF, which includes only the essential prime implicants necessary to cover all instances where the outcome is positive. The resulting logical expression can be further simplified to reduce the number of literals, though this is an NP-complete problem with various existing algorithms to address it.

Despite these improvements, there are still significant drawbacks to representing guidelines purely as simplified DNF's for each outcome. Firstly, expressions such as "Any $x$ of the following $y$ are true" require $\binom{x}{y}$ conjunctive literals to capture in pure formal logic. Secondly, these simplified DNF's fail to capture both implicit and explicit dependencies between predicates. (For example, "Does the car have $>20\%$ fuel?" has an explicit dependency on the question "Does the car use fuel?"). Thirdly, they do not provide an evaluation order for predicates, assuming that the truth values of all predicates are known simultaneously. However, in most real-world applications, the truth values of certain predicates determine which predicates need to be evaluated next. More importantly, these predicate determinations are often expensive, especially when a human is involved in evaluating them.

To address these limitations, we propose a novel symbolic representation called GDT (Guideline Decision Trees). GDT combines formal logic and decision trees to retain the expressive power of pure formal logic while overcoming the aforementioned drawbacks. A GDT is a directed acyclic graph comprising constructs such as predicates, outcomes, aggregators, and directed edges that represent logical relationships and evaluation sequences. This structure allows GDT to effectively capture complex guideline logic, dependencies, and evaluation orders.

### GDT Schema

A GDT (GDT) is a directed acyclic graph that encompasses various types of nodes and edges, each serving a distinct role in the decision-making process.

Nodes in a GDT include Predicates, Aggregators, and Outcomes. Predicates are independently verifiable truth statements about the world at a given moment. To ensure generalizability and simplicity, these predicates are modeled without being associated with specific entities. Aggregators, on the other hand, are nodes that combine the results of one or multiple predicates into a single outcome. For instance, they can represent conditions like “at least one is true” or "exactly two are false." These aggregators have specific properties: a condition that specifies the type of comparison ($\leq$, $==$ , $\geq$), a condition count which is an integer representing the number for the comparison, and a type indicating whether the condition is positive or negative. Outcomes, representing the possible results of executing the GDT, are all positioned as leaf nodes within the graph.

Edges in a GDT can be classified into directed edges and feeder edges. Directed edges connect nodes and signify logical relationships and evaluation sequences. A directed edge, denoted as `<source> -> <target>, <condition>`, implies that the target node should be evaluated if the source node meets the specified condition (True or False). Additionally, it indicates an "AND" relationship in all paths that include this edge. For example, the edges `Predicate_0 -> Predicate_1, True` and `Predicate_1 -> Outcome_0, False` imply that Outcome_0 is reached when Predicate_0 is true and Predicate_1 is false.

Feeder edges, represented as `<source>` $\rightarrow$ `<aggregator>`, indicate that the aggregator node’s determination depends on the output of the source node. These edges enable aggregators to collect and combine inputs from multiple sources.

In a GDT, an outcome path is defined as any path that begins at a root node (an aggregator or predicate with no incoming edges) and ends at an outcome. Conversely, an intermediate path is one that starts from a root node but does not end at an outcome. These paths illustrate the flow of logic from initial conditions to final determinations, showcasing how various predicates and aggregators interact to produce outcomes.

An example of a GDT is illustrated in Figure 3, highlighting how these components interact to model guidelines effectively and support automated decision processes.

### Methods

Every form of guideline that maps predicates to outcomes at a given instant in time can be represented in a GDT. It is important to note that multiple valid GDTs can exist for a single text-based guideline representation.

To transform free-text guidelines into GDTs, we employ both baseline models with few-shot prompting and task-specific fine-tuned models. Specifically, we use GPT-4 and GPT-3.5-turbo for baseline evaluations, while creating fine-tuned versions of GPT-3.5-turbo and LLAMA-3 8b. Converting free-text guidelines into accurate GDTs is a particularly challenging task for LLMs. It involves not only extracting the correct predicates and outcomes but also identifying the correct relationships between them using aggregators and edges. Evaluating the correctness of a generated GDT against a ground-truthed version is also challenging since two correct GDTs could have differently worded predicates and outcomes. Preliminary evaluation results indicate that the predicates generated by baseline models almost always differed from ground truth predicates both in number and meaning.

Therefore, we further break down the task of converting free text to GDT into subtasks, as shown in Table 1. We evaluate both baseline and fine-tuned model performance on each of these tasks. To train fine-tuned models on these specific tasks, we created a dataset comprising free text guidelines and their corresponding GDT representations. Since no such dataset was readily available and human annotation is expensive, we developed a novel method to synthetically generate data. We created randomly generated GDTs and performed a series of transformations on them, including adding synthetic noise to create corresponding free-text representations.

Additionally, we generated intermediate DNF form representations, unenriched summaries, and extracted predicates to provide a comprehensive dataset for the subtasks mentioned in Table 1. To facilitate the evaluation of GDTs generated by LLMs, we created library functions to parse textual representations of GDTs and methods to compare them against ground truth versions. Since multiple

correct GDTs can exist for each guideline, we used the logical equivalence rate as a proxy for correctness. Furthermore, we measured the delta between edges-to-nodes ratio as a measure of the complexity of two GDTs.

Our findings are presented in the Results and Evaluation section, highlighting the effectiveness and challenges of our approach.

# 5 Experiments

## 5.1 Data

Converting free-text into GDT is a challenging task, but the inverse task—converting GDTs into free-text—is simpler. We used this idea to create a five-step translation pipeline that generates random GDT's and converts them into free-text representations.

First, we programmatically generate random GDT skeletons with various combinations of predicates, aggregators, and outcomes. For our experiments, we used a dataset of trees with an average height of 4 and an average width of 4, each containing 2 outcomes. The algorithm ensures the generation of valid GDT's, enforcing additional constraints on edge creation to produce guidelines that closely resemble real-world examples. The exact algorithm for this generation process is outlined in A.1

Next, we generate fake predicates and outcomes. For each GDT skeleton, we use GPT-4 to imagine a random scenario and create dummy predicate and outcome statements corresponding to that scenario. A high temperature value for GPT-4 ensures greater creative variance in scenario generation, mitigating the risk of overfitting to specific types of scenarios.

Following this, we enrich the GDT skeletons by mapping each predicate and outcome to one of the generated statements. This step results in a complete GDT with contextually relevant predicates and outcomes.

To transform the GDT into a free-text representation, we first convert it into a structured text representation. We utilize two important properties of the GDT: any path in the GDT corresponds to conjunctions of predicates (or their negations, as indicated by edge conditions) along that path, and multiple incoming paths to a node represent disjunctions of possible paths leading to the node. This allows us to transform the GDT into a textual minimal DNF form representation, which is a deeply nested bulleted list of all outcome paths.

Once we have a list of all outcome paths, we use GPT-4 to convert this into a free-text paragraph summary, ensuring that all relevant information is captured. However, due to the uncertain behavior of LLMs, this step can introduce potential loss of information. Qualitative analysis indicates that roughly 5% of the time, some information from the GDT is lost, hallucinated, or misrepresented during this translation step.

To further process the summaries, we instruct a cheaper LLM, GPT-3.5-turbo, to add irrelevant words, phrases, or sentences to the summaries that do not affect the logical information contained within them. This adds a layer of noise to the free-text summaries.

We store the GDT objects along with the intermediary translations in a dataset for evaluation, training, and fine-tuning on various tasks. It is important to note that the generated summaries often contain nonsensical guidelines. For example, an example statement in the summary might be: "The car must be driven at full speed if either the car's battery is low or the driver has had McDonald's for lunch." Creating such illogical mappings is an intentional design choice, forcing the LLM to focus on the logical mapping between predicates and outcomes rather than relying on its parametric knowledge of the world.

# Experiment Details

To achieve the primary objective of creating a system that can convert free-text guideline representations to summaries, we devised the following three tasks to measure and improve performance on.

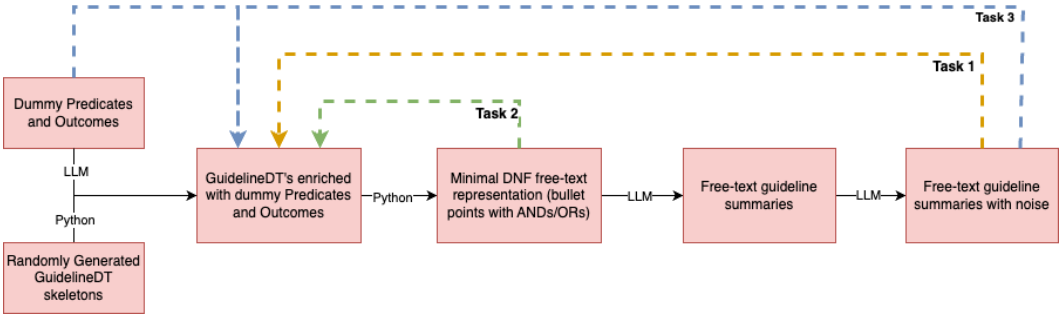


Figure 2: a) Synthetic data generation pipeline b) Task inputs and outputs

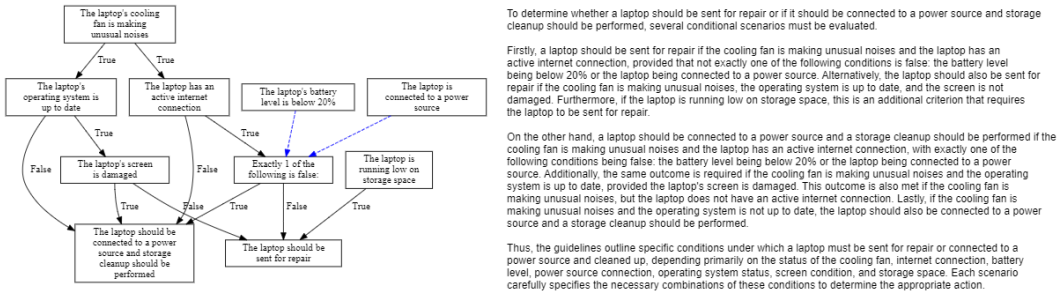


Figure 3: Example of a synthetically generated GDT and its corresponding free-text guideline representation

### Tasks

**Task 1: GDTs from Guideline Summaries** This is the primary task, focusing on converting guideline summaries directly into GuidelineDTs (GDTs).

**Task 2: GDTs from Guideline Minimal DNF Form** This task involves converting minimal DNF forms of guidelines into GDTs. We posit that it is worth evaluating performance on this task since a large portion of guidelines often follow this format.

**Task 3: GDTs from Guideline Summaries + Extracted Predicates and Outcomes** Tasks 1 and 2 show poor performance on baseline model evaluations primarily due to failures in the predicate extraction step. Often, outputs from these models merge two predicates into one or incorporate aggregator/negation logic within the extracted predicates. To address this, we propose this task to measure the performance of deducing logical mappings when outcomes and predicates are already provided along with free text.

### Evaluation Methods

We measure the performance on these tasks using the GPT-4o, GPT-3.5-turbo and LLAMA 3 - 8b language models.

Since GDTs are novel structures and large language models have not been trained on them, we provide a thorough definition of a GDT and illustrate examples of how to represent a GDT in plain text in the prompts for baseline model evaluations. Additionally, we use few-shot prompting with example input and output pairs for each task. However, we do not instruct the models on the algorithm

for creating GDTs from free-text, as multiple algorithms exist and they are challenging to describe in plain text. Our aim is for the model to learn these algorithms in an unsupervised manner.

### Supervised Fine-Tuning

To improve model performance, we employ supervised fine-tuning (SFT), which is recommended for improving task-specific performance when instructions are difficult to describe in plain text.OpenAI We fine-tune GPT-3.5-Turbo and LLAMA 8B on the three tasks. For GPT, we use the fine-tuning API provided by OpenAI. To fine-tune LLAMA 8b, we utilize the Unsloth library to perform Parameter Efficient Fine-Tuning (PEFT) using Low-Rank Adaptation (LORA) on a T4 GPU. Notebook code for fine-tuning LLAMA is provided in the project repo.

For fine-tuning we use a training/validation set of 500/50 input-output pairs. By fine-tuning these models, we aim to enhance their ability to convert free-text guidelines into accurate GDTs, thereby improving the overall performance on the specified tasks.

## 5.2 Results

The results show that baseline models consistently perform poorly on all three tasks. This is in line with the initial hypothesis that foundational models would struggle to create GDT's because they have never seen them during training and because ICL based prompting is not sufficient to teach a model how to construct GDT's. For simple enough GDT's (<5 nodes), we still see a reasonably good performance in accuracy and logical equivalence rate. As the number of nodes increase, logical equivalence rates go close to 0. Baseline models perform better on Task 2 > Task 3 > Task 1. This result follows the order of simplicity of the three tasks.

Although we fine-tuned a LLAMA 3 8B model on the same dataset, we omit the metrics here for both the baseline and fine-tuned model evaluations since they both yielded less than 5 perfect matches across all tasks.

The fine-tuned GPT-3.5 model demonstrates considerably improved performance across the three tasks.

Table 1: Performance Metrics for Tasks. Test set: n = 150

| Task + Inputs | Model / Metric | Perfect Matches | Logical Equivalence Rate | Edge-to-node ratio delta |
|---|---|---|---|---|
| T1: Free-Text | GPT-4o | 4% | 35% | 0.0 |
| | GPT-3.5 (FT) | **32%** | **67%** | 0.0 |
| T2: Minimal DNF Form | GPT-4o | 23% | 40% | 0.0 |
| | GPT-3.5 (FT) | **90%** | **95%** | 0.0 |
| T3: Free-text, Predicates, Outcomes | GPT-4o | 18% | 53% | 0.0 |
| | GPT-3.5 (FT) | **66%** | **81%** | **-0.27** |

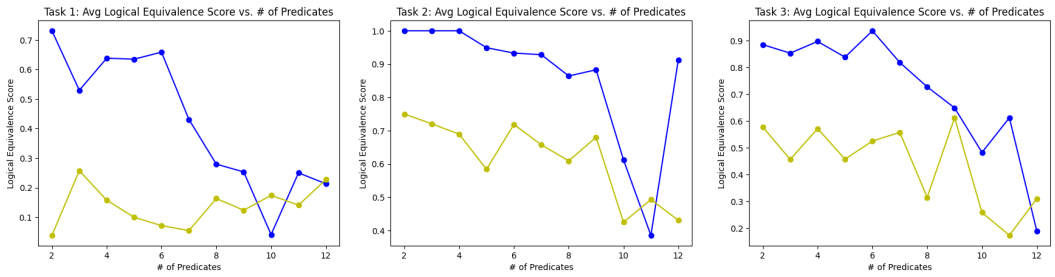


Figure 4: Logical Equivalence Score Avg vs no. of Predicates. Yellow line: Baseline Model. Blue line: Fine-Tuned model

## 6 Analysis

The evaluation mechanisms used for Task 1 and Task 2 are notably strict and only provide a lower bound on performance. This is primarily because they strongly penalize even the slightest mismatch in predicate and outcome extraction. A qualitative analysis of outputs from Task 1 reveals that when predicates and outcomes do not align with the ground truth, it is often due to multiple predicates being grouped together. This penalization highlights a flaw in the evaluation mechanism rather than the generation itself.

Consequently, results from Task 3—where predicates and outcomes are already provided to the LLM—offer better indicators of the system's ability to understand logical mappings in real-world scenarios. However, the quantitative results alone do not tell the entire story.

A qualitative analysis of real-world medical guidelines with the model fine-tuned on Task 1 yields mixed results. Although the generated GuidelineDT (GDT) does not contain incorrect mappings, it fails to capture many of the possible outcome paths. In its current state, the model struggles to fully generalize to medical guidelines. I suspect the root cause lies in the Disjunctive Normal Form (DNF) to summary step of the synthetic data generation process. This step has two failure modes: (a) loss of information during translation, and more importantly, (b) all generated summaries exhibit very similar "ChatGPT-like" verbiage. Since an LLM with the same prompt is used to generate all summaries, they tend to read the same way, regardless of the subject matter. This uniformity fails to capture the varied verbiage and styles found in real-world summaries. A potential future step would be to introduce variance in summary styles using different LLM system prompts.

An interesting and surprising finding is that there is almost no difference in complexity, measured by the edge-to-node ratio, between generated GDTs and ground truth GDTs. Upon closer inspection, almost all GDT pairs that are logically equivalent also share the same structure. This suggests that the LLM-generated summary styles are closely coupled with the underlying GDT structures, potentially causing the models to overfit to these types of summaries. This observation indicates that while the current approach has strengths, there is room for improvement in making the model more adaptable to the diversity found in real-world guidelines.

## 7 Conclusion

As part of this project, we discussed the need for symbolic guideline representations and explored various forms of guideline representations. We proposed a novel schema - GDT, to capture all kinds of guidelines in a succinct way. We aimed to create a system that can create GDT's based on free-text guidelines and devised corresponding tasks to evaluate performance on. To create this system, we started with creating a dataset using a synthetic data generation technique, without the need for human annotation. We then trained a first version of this system using this dataset, and demonstrated effective performance gains over baseline models. We recognized limitations in our data generation and evaluation mechanisms.

There is plenty of work to be done for automatically generated GDT's to have practical relevancy. Apart from the work required to address the discussed limitations, a potential unlock to better GDT generation would be to experiment with using logical equivalency rates directly as a loss function during supervised fine-tuning instead of cross entropy on tokens. We also intend to experiment with fine-tuning on smarter models like GPT-4 and Claude.

## 8 Ethics Statement

The societal and ethical impact of using AI to automate decision-making varies significantly depending on the domain of application. For this project, we focused on the guideline-evidence-to-outcome mapping task, which is particularly prevalent in the insurance industry. It is important to acknowledge that, especially in healthcare, the insurance industry already suffers from a negative public image. Conflicts between insurance companies, patients, and hospitals over treatment payments are common, and these companies often have the final say in decisions that can lead to significant monetary losses for patients.

The introduction of an automated decision-making tool in this context could exacerbate issues of accountability within the insurance industry. It is crucial to ensure that human reviewers sign off on AI-made decisions, as human lives and finances are at stake. Such automated tools should be used as productivity aids for critical decision-makers rather than substitutes. This approach helps maintain accountability and ensures that ethical considerations are appropriately addressed in the decision-making process.

Mitigation strategies include:

Human Oversight: Ensuring that all AI-driven decisions are reviewed and approved by human experts to maintain accountability and accuracy. Transparency: Making the decision-making process of AI systems transparent and understandable to all stakeholders, including patients, healthcare providers, and regulators. Bias Mitigation: Continuously monitoring and addressing potential biases in AI models to ensure fair and equitable treatment of all individuals. Regulatory Compliance: Adhering to existing regulations and guidelines to protect patient rights and ensure ethical use of AI in decision-making. By implementing these strategies, we can mitigate the ethical challenges and societal risks associated with using AI for decision-making in sensitive domains like insurance.

## References


Binbin Li, Tianxin Meng, Xiaoming Shi, Jie Zhai, and Tong Ruan. 2023. Meddm:llm-executable clinical guidance tree for clinical decision-making.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

OpenAI. Fine-tuning, openai api.

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482.


## A Appendix

### A.1 Algorithm for Generating Random Guideline Decision Tree

**Algorithm 1** Generate Random Guideline Decision Tree (GDT) by Level

**Input:** max_width, max_height, outcome_types
**Output:** GuidelineDT
$level \leftarrow 1$
$outcomes \leftarrow$ [Outcome($name =$ f"Outcome_" $+ i$) for $i$ in range($outcome\_types$)]
$prev\_level\_nodes \leftarrow [outcomes[:]]$
$directed\_edges \leftarrow []$
$feeder\_edges \leftarrow []$
$predicates \leftarrow []$
$aggregators \leftarrow []$
**while** $level < max\_height$ **do**
    $level\_width \leftarrow$ random.randint$(1, max_width)$
    $nodes \leftarrow []$
    **for** $i \leftarrow 1$ to $level\_width$ **do**
        $node\_type \leftarrow$ random.choices(["$Predicate$", "$Aggregator$"], $[4, 1], 1)[0]$
        **if** $node\_type ==$ "Predicate" **then**
            $predicate \leftarrow$ Predicate($name =$ f"Predicate_" + len($predicates$))
            nodes.append(predicate)
            predicates.append(predicate)
        **else**
            $aggregator \leftarrow$ CREATE_AGGREGATOR(aggregators, predicates, feeder_edges)
            nodes.append(aggregator)
        **end if**
    **end for**
    **for** $source$ in nodes **do**
        $level\_choices \leftarrow$ random.choices($range$(len($prev\_level\_nodes$)), 2)
        $marked\_nodes \leftarrow$ set()
        **for** $condition, level\_choice$ in zip($[True, False], level\_choices$) **do**
            $potential\_targets \leftarrow prev\_level\_nodes[level\_choice]$
            **if** $level\_choice == 0$ **then**
                $k \leftarrow 1$
            **else**
                $k \leftarrow$ LEFT_SKEWED_INTEGER_SAMPLE(1.0, len(potential_targets))[0]
            **end if**
            $targets \leftarrow$ random.sample($potential\_targets, k$)
            **for** $target$ in targets **do**
                **if** $target$ not in marked_nodes **then**
                    directed_edges.append(DIRECTEDEDGE(condition, source, target))
                    marked_nodes.add(target)
                **end if**
            **end for**
        **end for**
    **end for**
    prev_level_nodes.append(nodes)
    $level \leftarrow level + 1$
**end while**
**return** GUIDELINEDT(predicates, aggregators, outcomes, directed_edges, feeder_edges)