

T2ER: A Chinese Benchmark for Text-to-Executable-Rule Generation in Medical Insurance

Hanjing Yan¹, Rubin Li¹, Yuxuan Lin², Siyi Zhu¹, Yuxiang Wang¹, Guangya Yu¹,
Yuxiang Chu¹, Yongqi Fan^{1,3}, Yinan Wu¹, Bao Wei⁴, Weiyan Zhang^{1*}, Tong Ruan^{1*}

¹ East China University of Science and Technology, Shanghai, China

² Fudan University, Shanghai, China

³ Tencent, ⁴ Hainan Cancer Hospital, Hainan, China

Y21250025@mail.ecust.edu.cn, {weiyanzhang, ruantong}@ecust.edu.cn

Abstract

Medical insurance plays a pivotal role in the healthcare system, where effective supervision relies on the precise execution of complex medical insurance rules. Existing solutions for medical insurance violation detection mainly rely on costly manual labor, limiting efficiency. Although current Large Language Models (LLMs) have demonstrated significant potential in general Natural Language Processing (NLP) tasks, they still struggle to understand these rules, and a corresponding evaluation benchmark is lacking. Hence, we propose T2ER, a benchmark to evaluate the capability of LLMs in converting texts into executable rules in medical insurance. T2ER consists of 2,115 rules and 5,636 synthetic Electronic Medical Record (EMR) test cases for execution verification. Furthermore, we design a dual evaluation framework spanning rule and execution levels to comprehensively assess transformation quality. We conduct extensive experiments on advanced LLMs. The results reveal a significant human-AI performance gap, indicating that this conversion remains a formidable challenge. The dataset is available at <https://github.com/YHJ1223/T2ER>.

1 Introduction

The basic medical insurance system is a fundamental institutional arrangement that pools health risks and reimburses the medical expenses of the insured. As a cornerstone of the social security framework, it safeguards public health rights and underpins social stability (Li et al., 2022).

However, medical insurance violations are widespread and severe, yet remain difficult to detect. Concealed and diverse violations such as Duplicate Charges, Excessive Charges, and Ineligible Drug Use are prevalent. These violations not only result in the waste of valuable medical resources but also severely infringe upon the rights of the

insured, posing a substantial threat to the long-term sustainability of the medical security system. Currently, detecting such violations largely relies on manual inspection. The underlying rules are scattered across policy documents at different administrative levels, clinical practice guidelines, and internal hospital regulations, exhibiting pronounced characteristics of unstructured text, cross-document dependencies, and cross-level heterogeneity. In practice, existing systems typically require medical insurance experts and engineers to manually translate these natural-language rules into regular expressions, rule engines, or structured query language (SQL) functions. While this predefined logic-based matching paradigm offers strong interpretability, it incurs substantial maintenance costs and human effort when confronted with frequent policy updates, significant regional variations, and complex hospital-specific configurations.

Large language models (LLMs) offer a promising direction toward automating medical insurance violation detection. LLMs have shown remarkable potential in medical knowledge representation, complex clinical reasoning, and structured modeling (Singhal et al., 2023; Thirunavukarasu et al., 2023), and researchers have recently begun applying them to administrative tasks such as medical quality control indicator calculation (Yu et al., 2025). In practice, violation detection requires matching each patient record against a set of rules and determining whether any constraint is breached. Rather than performing this matching directly, LLMs are well suited to parsing unstructured rule texts into structured, executable logic—selecting the appropriate checks, extracting their parameters, and composing them into verifiable expressions—thereby bridging the semantic understanding of natural language with the high-efficiency execution of traditional rule engines.

Nevertheless, directly applying LLMs to record-by-record “questioning-style” inference over mil-

*Corresponding Authors.

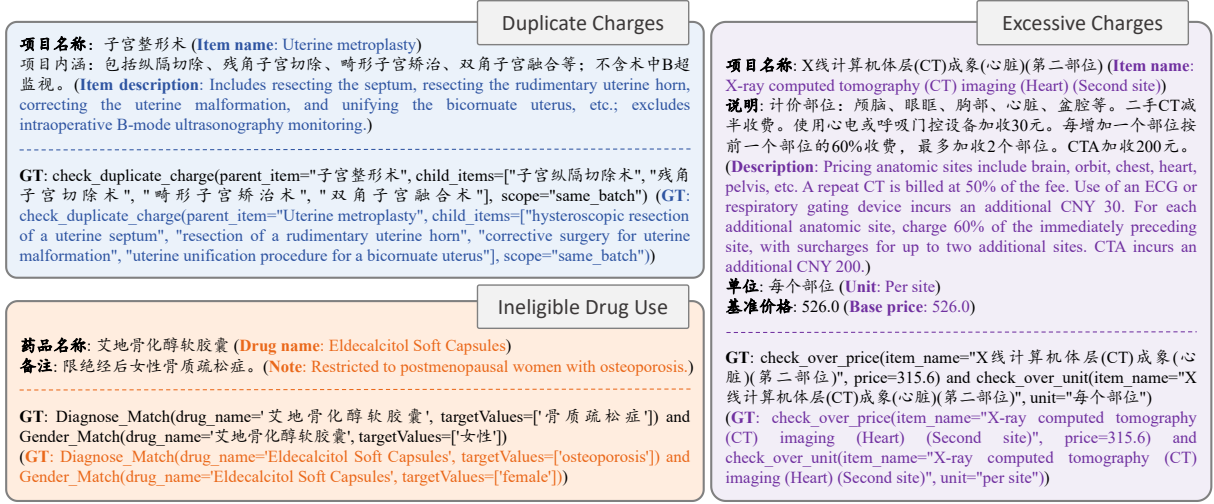


Figure 1: Examples of T2ER in three violation categories (Duplicate Charges, Ineligible Drug Use and Excessive Charges). Each case includes natural language text inputs and ground truth (GT).

lions of patient records incurs prohibitive computational costs and raises serious privacy concerns. A more practical paradigm is to use LLMs to convert natural-language rules into executable functions that can be efficiently executed by rule engines. However, existing benchmarks, as shown in Table 1, focus primarily on clinical decision-making and question answering, and offer no means to evaluate this rule-conversion capability. To fill this gap, we construct **T2ER (Text-to-Executable-Rule)**, a benchmark designed to evaluate models' ability to convert natural-language medical insurance rule texts into executable rules. T2ER covers three major categories of core medical insurance violations: duplicate charges, ineligible drug use, and excessive charges. Furthermore, we design a dual evaluation framework based on abstract syntax tree (AST) structural matching and consistency of downstream execution results, enabling comprehensive assessment of model performance at both the rule and execution levels. Experimental results indicate that current LLMs still face substantial challenges in generating executable medical insurance rules: although models exhibit a basic capability for planning the overall logical structure, they show significant deficiencies in precise parameter alignment and compositional constraint handling, highlighting the capability bottleneck of LLMs in rigorous vertical domains.

In summary, our contributions are as follows:

- We propose T2ER, the first comprehensive benchmark for evaluating models' ability to translate natural-language medical insurance

rule texts into executable rules.

- We design a dual evaluation framework combining AST structure matching with execution consistency to comprehensively assess rule generation quality.
- Extensive experiments are conducted on 12 representative LLMs. The results indicate that current LLMs exhibit limitations in generating executable medical insurance rules.

2 T2ER Task Formulation

The task aims to convert natural-language rule texts into executable rules for medical insurance violation detection. Formally, it consists of two stages: Rule Generation and Execution-based Detection.

In the Rule Generation stage, given a natural-language rule $\mathcal{T}$ and a predefined function bank $\mathcal{B}$, the model Φ is required to generate a syntactically valid executable expression R with instructions $\mathcal{I}$. This process can be formulated as:

$$R = \Phi(\mathcal{T}, \mathcal{B} \mid \mathcal{I}), \quad (1)$$

where R consists of multiple atomic functions from $\mathcal{B}$ combined using Boolean operators (AND/OR/NOT).

In the downstream execution phase, the generated rule R is executed on a corresponding EMR $\mathcal{E}$ to produce the structured execution result $\hat{\mathcal{O}}$:

$$\hat{\mathcal{O}} = \text{Exec}(R, \mathcal{E}) = (v, r), \quad (2)$$

where $v \in \{\text{True}, \text{False}\}$ indicates whether a violation occurred and r is the specific violation reason.

Dataset	Domain		Task		Environment & Facility			
	Med	Zh	Code	Audit	Exec	EMR	Schema	Explain
CMB (Wang et al., 2024)	✓	✓	✗	✗	✗	✗	✗	✗
CMEExam (Liu et al., 2023)	✓	✓	✗	✗	✗	✗	✗	✓
MedBench (Cai et al., 2024)	✓	✓	✗	✗	✗	✗	✗	✗
MedEval (He et al., 2023)	✓	✗	✗	✗	✗	✗	✗	✗
CMQCIC-Bench (Yu et al., 2025)	✓	✓	✗	✓	✗	✓	✗	✗
MedCalc-Bench (Khandekar et al., 2024)	✓	✗	✗	✗	✗	✓	✗	✓
EHRSQL (Lee et al., 2022)	✓	✗	✓	✗	✓	✓	✓	✗
EHRNoteQA (Kweon et al., 2024)	✓	✗	✗	✗	✗	✓	✗	✗
EHRSHOT (Wornow et al., 2023)	✓	✗	✗	✗	✗	✓	✗	✗
MedSafetyBench (Han et al., 2024)	✓	✗	✗	✗	✗	✗	✗	✗
Text-to-ES Bench (Xue et al., 2025)	✗	✗	✓	✗	✓	✗	✓	✗
T2ER	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of the T2ER benchmark with representative benchmarks. Med, Zh, Code, Audit, Exec, EMR, Schema and Explain represent medical domain, Chinese, executable code generation, compliance or violation detection, execution-based evaluation, evaluation on patient records, schema-constrained generation, and reason annotation, respectively.

3 Data Construction

In this section, we detail the construction of the T2ER benchmark, as shown in Figure 2, which comprises function bank construction, executable rule annotation, and patient case synthesis, with multi-level quality control applied throughout.

3.1 Function Bank Construction

To convert natural-language texts into executable rules, we construct a domain-specific function bank $\mathcal{B}$. Based on real-world medical-insurance violation reports, we first manually define the violation types and triggering conditions, as well as the function name of each of the 11 atomic functions (e.g., `check_duplicate_charge`, `Diagnose_Match`). We then employ Claude-4.5-Sonnet to complete the remaining schema fields, including each function’s description, parameter, and type. Finally, domain experts review and revise every entry to produce the final function bank $\mathcal{B}$ (detailed in Appendix D.1).

3.2 Executable Rule Annotation

Source Data Processing. We collect natural-language medical insurance rules from structured and semi-structured sources (i.e., Medical Service Price Item Specifications and National Basic Medical Insurance Drug Catalogs) and convert them into structured JSON using OCR and table parsing. We then filter the fields that carry core rule semantics (item name, definition, instructions/notes, price, etc.) and classify them into three categories based on medical-insurance cost-control logic: (1) *Duplicate Charges*, which involves mutually exclusive

billing within the same batch caused by parent-child item inclusion relationships; (2) *Excessive Charges*, which involves standard price-calculation and unit constraints; and (3) *Ineligible Drug Use*, which involves medication restrictions regarding indications, age, gender, symptoms, and time ranges.

Annotation Process. We map the structured texts to executable logical expressions based on the function bank $\mathcal{B}$, a process that constitutes a complex constrained semantic parsing task. To enhance consistency and controllability, we explicitly decompose rule annotation into three steps. (1) Function Selection: identifying the violation mechanism implied by the text and selecting the appropriate set of atomic functions. (2) Parameter Extraction & Reasoning: extracting parameter values from natural language, which involves multiple reasoning tasks, including arithmetic reasoning (computing values from calculation rules in the descriptions) and medical-term standardization (mapping colloquialisms, abbreviations, or non-standard expressions to standard terms). (3) Logic Composition: using Boolean operators (AND/OR/NOT) to combine multiple function calls into the correct logical structure expressing multi-condition composite constraints.

Collaborative Annotation. To ensure annotation quality, we adopt a three-stage workflow. First, LLMs generate the initial rule code. Next, 12 student annotators, including medical students and computer science students who received physician-led, task-specific training in medical terminology and medical-insurance rule annotation, verify the selected functions, parameter consistency, seman-

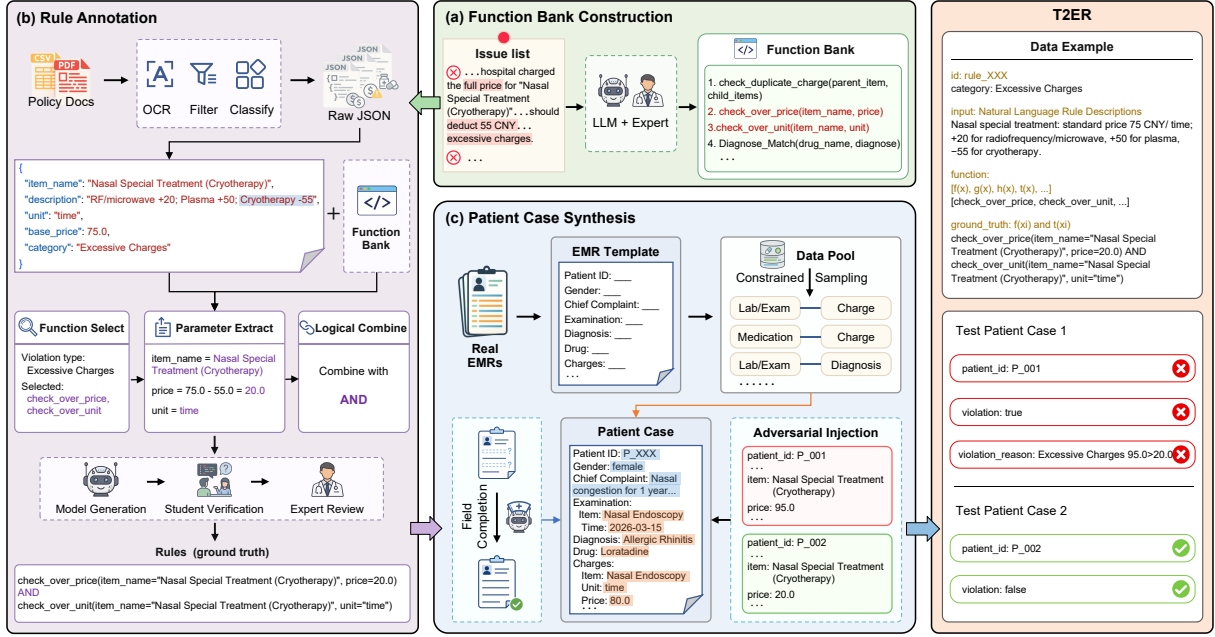


Figure 2: The T2ER benchmark construction consists of three stages: (a) function bank construction through manual semantic specification, LLM-assisted schema completion, and expert refinement; (b) rule annotation, which processes source policy documents and maps them to executable rules through function selection, parameter extraction, and logical composition; and (c) patient case synthesis, where constrained sampling and adversarial injection produce execution-labeled test cases.

tic alignment, and Boolean composition. Each rule is independently verified by two student annotators, and disagreements are resolved by three physicians. Each rule is additionally validated by automated scripts for syntactic conformance and compliance with the parameter schemas defined in $\mathcal{B}$. This process yields 2,115 $(\mathcal{T}, R)$ rule pairs. The three physicians then independently validate the final dataset, achieving a Fleiss’ kappa of 0.82, which indicates almost perfect agreement (Fleiss, 1971; Landis and Koch, 1977).

3.3 Patient Case Synthesis

EMR Synthesis and Adversarial Injection. To enable execution-based evaluation, we synthesize 2,000 structured patient EMRs. We first use a small set of real de-identified hospital EMRs (Appendix D.2) to define a seven-module template covering basic information, medication information, laboratory reports, examination reports, surgical records, billing reports, and diagnoses. Based on the drug catalogs, we design field-correlation rules (e.g., “Examination-Billing” consistency) and build foundational data pools for drugs, examinations, and billing items. We then generate structured core fields via constrained sampling, followed by LLM completion of weakly structured fields (e.g., his-

tory of present illness) under these constraints, and perform consistency checks and revisions to obtain foundational EMR samples with complete structure and coherent medical logic. On top of these, we apply a rule-guided adversarial-injection strategy to construct a test set covering compliance, violation, and boundary cases. Guided by each rule, we inject the items it governs into the corresponding EMR fields with carefully chosen values to instantiate the three case types: compliance cases populate the fields with rule-conforming values, violation cases inject a targeted breach (e.g., a charged price exceeding the standard, or co-billing a parent and child item), and boundary cases assign a value adjacent to the rule’s decision threshold to probe its decision margin. For each rule, we generate or select multiple patient EMRs and execute the rule to obtain the expected output, recording whether a violation occurs (True/False) along with the violation type and reason. We further require that (1) each rule include both violating and non-violating cases, and (2) for multi-function rules, the dataset cover branching scenarios that enable independent verification of each sub-condition.

Execution-Based Quality Control. We verify the synthesized cases along two dimensions. First, an execution-consistency check runs all ground-

Metrics	Duplicate	Ineligible	Excessive	Total
# Rules	579	402	1,134	2,115
Avg. IT	38.91	52.10	73.17	59.78
Avg. OT	46.37	53.63	62.18	56.23
# Func. Calls	Single	Multiple	Multiple	-
# Test Cases	1,158	1,076	3,402	5,636

Table 2: Detailed statistics of the benchmark across different categories. IT, OT and Func. denote Input Tokens, Output Tokens and Function, respectively.

truth rules on their paired test cases, discarding any sample whose execution output conflicts with the expected violation label. Second, a branch-coverage analysis ensures that every atomic function and logical connective is exercised by both positive and negative samples. To further validate the complete dataset, the student annotators and physicians conduct a random-sampling review on 10% of all rule–case pairs, focusing on the accuracy of parameter-value extraction and the correctness of complex logical compositions. We enforce a per-batch pass rate of 98%; any batch falling below this threshold triggers a full revision, and the process repeats until quality stably converges to the gold standard.

4 Data Statistics

Benchmark statistics. As shown in Table 2, T2ER comprises 2,115 medical insurance rules and 2,000 patient EMRs, yielding 5,636 context-aware patient test cases for execution-based evaluation. The rules span the three most representative categories in medical insurance cost control, with Excessive Charges accounting for the largest share (53.6%), followed by Duplicate Charges (27.4%) and Ineligible Drug Use (19.0%). The paired natural-language inputs and executable rules have average token lengths of 59.78 and 56.23, respectively. All rules are grounded in a function bank $\mathcal{B}$ of 11 violation-detection functions spanning the three scenarios, where each function is specified by a natural-language description and a strict parameter schema covering required and optional parameters, type constraints, enumerations, and default values.

Diversity and compositional complexity. Beyond scale, T2ER poses substantial structural diversity in the target rules. To assess the difficulty of logic synthesis, we analyze the code structure of the ground-truth rules. Only 36.5% of the rules can be expressed with a single function call, while the remaining 63.5% require multi-function composition via Boolean operators (AND/OR/NOT) or

nested calls. This distribution indicates that simple one-to-one function mapping is insufficient, and that the benchmark demands compositional reasoning over multiple constraints.

5 Experiments

5.1 Experimental Models

We select 12 representative LLMs, covering both advanced open-source and closed-source models. The open-source models include Qwen3-8B/32B (Yang et al., 2025), DeepSeek-V3.2-Chat and DeepSeek-V3.2-Reasoner (DeepSeek-AI et al., 2025), and Baichuan-M2-32B (Dou et al., 2025). The closed-source models include Qwen3-Max (Yang et al., 2025), GPT-4o (OpenAI et al., 2024), GPT-5.2 (OpenAI, 2025), Gemini-2.5-Pro (Comanici et al., 2025), Gemini-3-Pro-Preview (Pichai et al., 2025), Claude-3.7-Sonnet (Anthropic, 2025a), and Claude-4.5-Sonnet (Anthropic, 2025b).

5.2 Experiment Setup

5.2.1 Evaluation Metrics

Following BFCL (Patil et al., 2025), we adopt a dual-perspective framework that assesses performance at the rule level and the execution level, via Overall Rule Accuracy (ORA), Binary Violation Accuracy (BVA), and Execution-Label Accuracy (ELA). Despite the task-specific names, these are standard measures: ORA is a strict AST exact-match metric (cf. logical-form exact match in semantic parsing), BVA a binary classification accuracy, and ELA an execution-level exact-match metric akin to functional correctness in DSL generation. Fine-grained metrics that decompose ORA and further characterize execution behavior are detailed in Appendix A. Throughout, $\mathbb{I}[\cdot]$ denotes the indicator function.

Rule level. ORA strictly evaluates whether the synthesized rule is structurally and semantically isomorphic to the ground truth. It is a composite of *Logic Structure Accuracy (LSA)* and *Function-Call Correctness (FCC)*, counting an instance as correct only when both hold:

$$\begin{aligned}
 \text{ORA} &= \frac{1}{N} \sum_{i=1}^N \mathbb{I}[\text{AST}(\hat{e}_i) \equiv \text{AST}(e_i)] \\
 &= \frac{1}{N} \sum_{i=1}^N \mathbb{I}[\text{LSA}_i = 1 \wedge \text{FCC}_i = 1],
 \end{aligned} \tag{3}$$

Model	Average			Duplicate Charges			Ineligible Drug Use			Excessive Charges		
	ORA	BVA	ELA	ORA	BVA	ELA	ORA	BVA	ELA	ORA	BVA	ELA
Open-source Models												
Qwen3-8B	26.76	72.44	43.58	9.15	60.89	29.62	21.39	70.41	30.69	49.74	86.01	70.42
Qwen3-32B	27.90	72.68	47.02	12.44	63.28	35.57	23.38	72.14	35.19	47.88	82.61	70.31
DeepSeek-V3.2-Chat	40.83	77.51	55.18	17.10	66.19	38.32	31.59	75.42	41.49	73.81	90.91	85.74
DeepSeek-V3.2-Reasoner	35.25	75.78	50.72	13.64	62.50	35.43	31.34	76.35	41.50	60.76	88.50	75.24
Baichuan-M2-32B	29.38	74.90	47.54	12.09	61.73	34.01	28.86	77.70	39.48	47.18	85.26	69.13
Closed-source Models												
Qwen3-Max	40.04	78.42	55.97	20.90	70.51	43.48	33.08	75.96	44.60	66.14	88.78	79.82
GPT-4o	36.82	76.20	50.75	13.99	64.15	35.45	29.35	75.00	37.64	67.11	89.45	79.17
GPT-5.2	41.97	79.13	56.44	15.54	67.91	39.88	31.09	75.95	40.90	79.28	93.53	88.53
Gemini-2.5-Pro	43.03	78.72	58.30	18.31	67.97	40.44	35.82	77.95	51.62	74.96	90.24	82.85
Gemini-3-Pro-Preview	<u>44.47</u>	<u>81.60</u>	<u>63.17</u>	19.86	69.72	43.65	<u>36.57</u>	81.66	60.15	76.98	93.41	85.71
Claude-3.7-Sonnet	40.78	76.72	56.49	<u>20.03</u>	<u>71.13</u>	42.52	35.57	78.58	48.92	66.75	80.45	78.03
Claude-4.5-Sonnet	47.02	82.22	63.22	19.34	72.04	40.31	37.06	<u>78.94</u>	60.22	84.66	95.67	89.13
Human Evaluation												
Human	94.67			91.00			95.00			98.00		

Table 3: Performance comparison of various LLMs on T2ER. We evaluate models using ORA (rule structural correctness), BVA (binary violation detection), and ELA (exact execution output match). **Bold** and underline denote the best-performing and second-best-performing models, respectively.

where $\equiv$ denotes AST isomorphism and $\hat{e}_i, e_i$ are the predicted and ground-truth expressions for instance i .

Execution level. Let $\hat{\mathcal{Y}}(x_i)$ and $\mathcal{Y}(x_i)$ denote the predicted and ground-truth violation sets for patient record x_i , derived from the execution output $\hat{\mathcal{O}} = (v, r)$ in Eq. (2). ELA, the primary execution metric, requires both the detected violations and their specific reasons to match exactly:

$$\text{ELA} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}[\hat{\mathcal{Y}}(x_i) = \mathcal{Y}(x_i)]. \quad (4)$$

BVA relaxes ELA to a binary classification task that checks only whether *any* violation is correctly identified. With $\hat{v}_i = \mathbb{I}[\hat{\mathcal{Y}}(x_i) \neq \emptyset]$ and $v_i = \mathbb{I}[\mathcal{Y}(x_i) \neq \emptyset]$:

$$\text{BVA} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}[\hat{v}_i = v_i]. \quad (5)$$

We further conduct a human evaluation on 500 samples as a complementary correctness baseline.

5.2.2 Implementation details

We uniformly employ zero-shot inference for all models, with the temperature set to 0.0 and the maximum output length set to 8,192 tokens. All other hyperparameters use their default values.

5.3 Main Results

Table 3 reports the performance of representative LLMs on the T2ER benchmark. It is observed that:

(1) Significant Human-AI Performance Gap.

A large gap remains between SOTA models and human experts. Even the strongest models achieve an ORA of only around 20% on the challenging Duplicate Charges task (e.g., 20.90% for Qwen3-Max), far below the human level of 91.00%. This indicates that current LLMs still struggle to generate executable medical insurance rules.

(2) Difficulty Stratification Across Task Types.

Model performance varies substantially across task categories. The average ORA for “Excessive Charges” reaches 66.27%, whereas it drops to 31.26% and 16.03% for the other two tasks. This confirms the variation in difficulty across different medical insurance violation tasks.

(3) Scaling Helps but Saturates Below the Performance Ceiling. Performance generally improves with model scale: within the Qwen series, the average ORA rises consistently from 8B (26.76) to 32B (27.90) and further to Max (40.04). However, this benefit quickly saturates—even the most advanced LLMs, such as Claude-4.5-Sonnet (47.02) and Gemini-3-Pro-Preview (44.47), plateau around 47% ORA and remain far below the human level of 94.67%. This suggests that parameter scaling alone is insufficient to close the human-AI gap on T2ER.

(4) Consistent Trends Between Rule-level and Execution-level Metrics. Models performing better on strict rule matching (ORA) generally also achieve better performance on execution-based metrics (BVA/ELA). This suggests that rule-

generation quality and execution performance are closely related and jointly reflect models’ capabilities on this task.

5.4 Analysis and Discussion

5.4.1 Fine-grained Error Analysis

Figure 3 shows a fine-grained AST-based error analysis of T2ER, from which we draw two observations.

Parameter grounding is the primary bottleneck. LSA is significantly higher than FCC across all models, indicating that models can plan the logical structure but struggle with strict function calling. Decomposing FCC further, FNM remains generally high—function selection is not the main difficulty—whereas PSC is strongly correlated with FCC (correlation coefficient ≈ 0.93), showing that the performance gap primarily stems from the parameter level. Detailed error examples are provided in Appendix C.

Stronger reasoning does not guarantee higher FCC. For instance, DeepSeek-Reasoner attains higher LSA and FNM but shows no advantage in FCC, suggesting that stronger generation capabilities may instead introduce more pronounced over-generation or parameter misalignment.

5.4.2 Analysis for Different Violation Types

Figure 4 breaks down the performance of six representative models, spanning open- and closed-source families, against rule complexity for each violation type.

Duplicate Charges. Figure 4(a) stratifies this task by the number of `child_items` into [1–2], [3–4], and [5+]. ORA decreases monotonically as the number of child items increases, with the average across the six models dropping from 18.20% to 14.52% and further to 7.02%. This consistent degradation indicates that current models have limited capacity to handle long, compositional rules.

Ineligible Drug Use. Figure 4(b) stratifies this task by the number of required function calls into [1], [2], and [3+]. Performance drops sharply as more calls are required, and the decline from one to two calls is especially steep: the average ORA plummets from 52.16% to 14.99%, then to 5.67% at three or more calls. This suggests that current LLMs struggle to coordinate multiple constraints within a single rule.

Excessive Charges. Figure 4(c) stratifies this task by arithmetic operation type: Add, Subtract, and Multiply. The models tend to perform better on

the additive category, yet performance does not follow a monotonic difficulty gradient. Error analysis attributes this to item-name extraction errors rather than the distinctiveness of the operation itself: such errors are pervasive across all operation types, as models exhibit a strong tendency toward “semantic simplification”, frequently truncating key qualifiers enclosed in parentheses. This reveals that the task poses the dual challenge of arithmetic reasoning and rigorous entity grounding.

5.4.3 Comparative Paradigm Analysis: Rule Exec vs. Direct NL

To verify the necessity of the T2ER task definition, we compared “Rule Execution” (Rule Exec) with “Direct Natural Language Inference” (Direct NL). As shown in Figure 5, Rule Exec demonstrates overwhelming advantages in both Execution-Label Accuracy and execution time: the average ELA improves by 24.04 percentage points, and inference speed achieves an acceleration of approximately 10^4 magnitude per sample. Therefore, the task of converting texts into executable rules proposed in this paper possesses significant practical relevance and necessity.

6 Related Work

6.1 LLMs in Medical Governance

LLMs have demonstrated strong clinical knowledge reasoning capabilities, achieving competitive performance on physician-licensing-style examinations and a broad range of medical benchmarks, as shown in Table 1. In Chinese clinical settings, comprehensive benchmarks support systematic evaluation and error analysis (Liu et al., 2023; Cai et al., 2024; Wang et al., 2024; Ouyang et al., 2024). As summarized in Table 1, existing benchmarks either generate request-specific executable queries (Xue et al., 2025; Lee et al., 2022) or directly produce patient-level decisions or calculation results (Yu et al., 2025; Khandekar et al., 2024). In contrast, T2ER converts policy texts into reusable executable rules and evaluates both rule-level correctness and execution outcomes. Medical instruction tuning and multi-agent collaboration further improve clinical alignment and robustness (Zhang et al., 2023, 2025; Tang et al., 2024). Beyond diagnosis-oriented tasks, LLMs are increasingly being evaluated in medical governance settings, including indicator computation and logic verification over structured records (Yu et al., 2025).

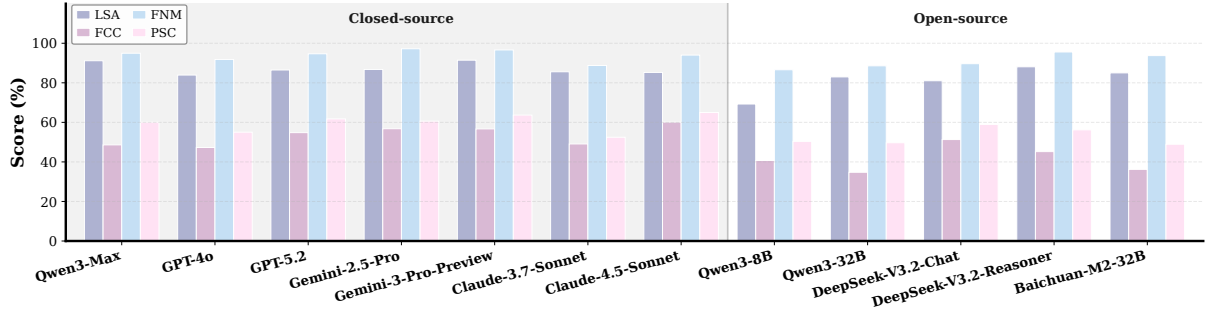


Figure 3: Fine-grained AST-based evaluation on T2ER for closed-source and open-source models. The figure reports LSA and FCC, as well as the FCC components FNM and PSC, across all evaluated models.

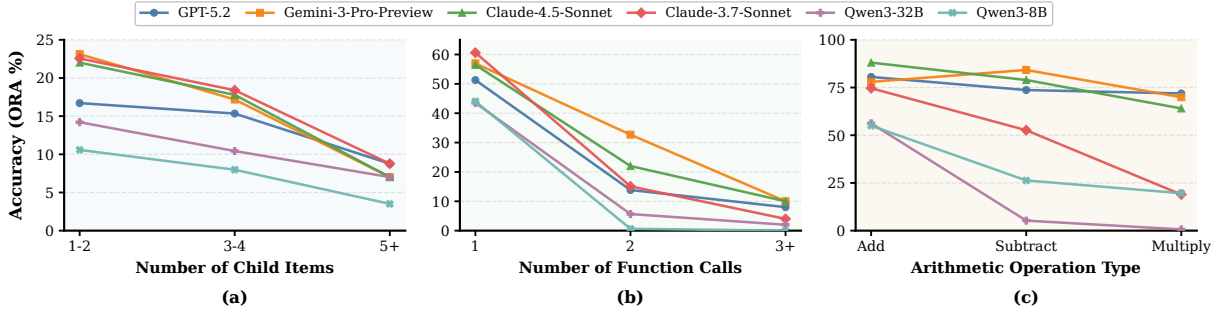


Figure 4: Performance breakdown by violation type and rule complexity on T2ER. (a) Duplicate Charges grouped by the number of child items (1–2, 3–4, 5+). (b) Ineligible Drug Use grouped by the number of required function calls (1, 2, 3+). (c) Excessive Charges grouped by calculation type (Add, Subtract, Multiply).

Recent work on health-insurance appeal adjudication has focused on classifying case-level outcomes from insurance case descriptions (Gartner, 2025). Together, these studies motivate our focus on converting natural-language rule texts into executable rules for medical insurance violation detection.

6.2 DSL Generation and Function Calling

Domain-specific language (DSL) generation maps natural language into executable formal representations, with evaluation focusing on syntactic validity, schema constraints, and execution consistency (Yu et al., 2018; Rabinovich et al., 2017; Xue et al., 2025; Yang et al., 2024; Kapur et al., 2025). Prior work has explored grammar-aware generation, abstract syntax modeling, constrained decoding, and iterative program synthesis to reduce structurally valid but semantically incorrect outputs (Scholak et al., 2021; Geng et al., 2024; Barke et al., 2024; Grand et al., 2024; Ugare et al., 2025; Mordechai et al., 2024). Closely related studies on structured function and API calling require models to select predefined interfaces and instantiate schema-constrained arguments to produce executable outputs (Li et al., 2023; Patil et al., 2024; Qin et al., 2024; Sbodio et al., 2024; Lopez et al.,

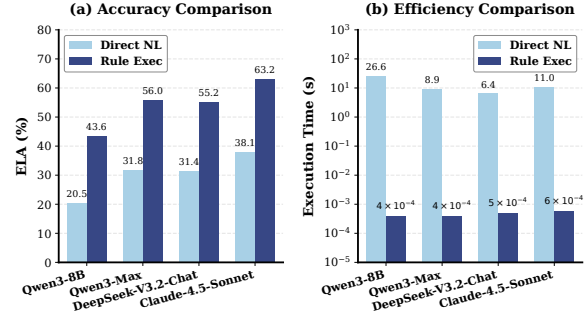


Figure 5: Comparison between Rule Exec and Direct NL on the T2ER benchmark. (a) Execution-Label Accuracy (ELA) across representative models. (b) Per-sample execution time in seconds (log scale) for the two paradigms.

2022). We formulate rule transformation as a DSL generation task using a constrained function-calling framework: natural-language rules are translated into executable rules based on a predefined function bank and parameter schema, with semantic correctness verified via execution on medical records.

7 Conclusion

This paper presents T2ER, the first comprehensive Chinese benchmark to evaluate LLMs’ ability to translate medical insurance rule texts into ex-

executable rules. The benchmark consists of 2,115 high-quality rules based on real-world policy documents, and builds a downstream evaluation environment with 2,000 EMRs and 5,636 test cases. We further propose a dual evaluation framework consisting of AST matching and execution consistency checks, which comprehensively evaluate rule quality from two dimensions. The experimental results on T2ER show that current LLMs have limited capability in performing the text-to-executable-rule task. Hence, the T2ER benchmark not only fills the gap in evaluating medical insurance rule transformation, but also provides critical empirical evidence for the subsequent development of highly reliable domain-specific LLMs.

Limitations

Our method has the following limitations:

- Our data annotation requires collaboration between humans and large models; currently, manual verification remains necessary, and we cannot yet rely entirely on LLMs for automated annotation.
- Constrained by the high barrier for expert verification and the volume of policy documents, the data scale is relatively moderate; however, this ensures high data quality and execution consistency.
- The T2ER benchmark focuses on the three most frequent violation categories rather than exhaustively covering all long-tail scenarios, as these categories constitute the vast majority of real-world medical insurance violations, thereby providing the most representative evaluation benchmark.

Ethical Considerations

Data Privacy. All rule texts are extracted from publicly released medical insurance policy documents. All EMRs in T2ER are fully synthetic and contain no personally identifiable information. Therefore, the released benchmark does not involve sensitive personal attributes or unauthorized data use.

Human Annotators. We directly invited 12 student annotators, comprising medical students and computer science students, based on their relevant academic backgrounds, as well as three physicians

from our collaborating hospitals. The computer science students received physician-led, task-specific training in medical terminology, medical-insurance policies, and rule-annotation guidelines before formal annotation. Participation was voluntary, and the total compensation for the annotation work was \$3,000.

Acknowledgments

This work is supported by the Shanghai Natural Science Foundation Project under Grant 25ZR1402116.

References

- Anthropic. 2025a. [Claude 3.7 Sonnet and Claude Code](#).
- Anthropic. 2025b. [Introducing Claude Sonnet 4.5](#).
- Shraddha Barke, Emmanuel Anaya Gonzalez, Saketh Ram Kasibatla, Taylor Berg-Kirkpatrick, and Nadia Polikarpova. 2024. [Hysynth: Context-free LLM approximation for guiding program synthesis](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 15612–15645. Curran Associates, Inc.
- Yan Cai, Linlin Wang, Ye Wang, Gerard de Melo, Ya Zhang, Yanfeng Wang, and Liang He. 2024. [Med-Bench: A large-scale chinese benchmark for evaluating medical large language models](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17709–17717.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Naveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3416 others. 2025. [Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities](#). *arXiv preprint arXiv:2507.06261*.
- DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. [Deepseek-v3. 2: Pushing the frontier of open large language models](#). *arXiv preprint arXiv:2512.02556*.
- Chengfeng Dou, Chong Liu, Fan Yang, Fei Li, Jiyuan Jia, Mingyang Chen, Qiang Ju, Shuai Wang, Shunya Dang, Tianpeng Li, Xiangrong Zeng, Yijie Zhou, Chenzheng Zhu, Da Pan, Fei Deng, Guangwei Ai, Guosheng Dong, Hongda Zhang, Jinyang Tai, and 14 others. 2025. [Baichuan-M2: Scaling medical capability with large verifier system](#). *arXiv preprint arXiv:2509.02208*.

- Joseph L. Fleiss. 1971. [Measuring nominal scale agreement among many raters](#). *Psychological Bulletin*, 76(5):378–382.
- Mike Gartner. 2025. [Health insurance coverage rule interpretation corpus: Law, policy, and medical guidance for health insurance coverage understanding](#). *Preprint*, arXiv:2508.03718.
- Saibo Geng, Berkay Döner, Chris Wendler, Martin Josifoski, and Robert West. 2024. [Sketch-guided constrained decoding for boosting blackbox large language models without logit access](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 234–245, Bangkok, Thailand. Association for Computational Linguistics.
- Gabriel Grand, Lio Wong, Maddy Bowers, Theo X. Olausson, Muxin Liu, Joshua B Tenenbaum, and Jacob Andreas. 2024. [Lilo: Learning interpretable libraries by compressing and documenting code](#). In *International Conference on Learning Representations*, volume 2024, pages 30399–30446.
- Tessa Han, Aounon Kumar, Chirag Agarwal, and Himabindu Lakkaraju. 2024. [MedSafetyBench: Evaluating and improving the medical safety of large language models](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 33423–33454. Curran Associates, Inc.
- Zexue He, Yu Wang, An Yan, Yao Liu, Eric Chang, Amilcare Gentili, Julian McAuley, and Chun-Nan Hsu. 2023. [MedEval: A multi-level, multi-task, and multi-domain medical benchmark for language model evaluation](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8725–8744, Singapore. Association for Computational Linguistics.
- Shreyas Kapur, Erik Jenner, and Stuart Russell. 2025. [Diffusion on syntax trees for program synthesis](#). In *International Conference on Learning Representations*, volume 2025, pages 41414–41432.
- Nikhil Khandekar, Qiao Jin, Guangzhi Xiong, Soren Dunn, Serina S Applebaum, Zain Anwar, Maame Sarfo-Gyamfi, Conrad W Safranek, Abid A Anwar, Andrew Zhang, Aidan Gilson, Maxwell B Singer, Amisha Dave, Andrew Taylor, Aidong Zhang, Qingyu Chen, and Zhiyong Lu. 2024. [MedCalc-bench: Evaluating large language models for medical calculations](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 84730–84745. Curran Associates, Inc.
- Sunjun Kweon, Jiyou Kim, Heeyoung Kwak, Dongchul Cha, Hangyul Yoon, Kwanghyun Kim, Jeewon Yang, Seunghyun Won, and Edward Choi. 2024. [EHRNoteQA: An LLM benchmark for real-world clinical practice using discharge summaries](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 124575–124611. Curran Associates, Inc.
- J. Richard Landis and Gary G. Koch. 1977. [The measurement of observer agreement for categorical data](#). *Biometrics*, 33(1):159–174.
- Gyubok Lee, Hyeonji Hwang, Seongsu Bae, Yeonsu Kwon, Woncheol Shin, Seongjun Yang, Minjoon Seo, Jong-Yeup Kim, and Edward Choi. 2022. [EHRSQL: A practical Text-to-SQL benchmark for electronic health records](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 15589–15601. Curran Associates, Inc.
- Jie Li, Qiaoling Lan, Enya Zhu, Yong Xu, and Dan Zhu. 2022. [A study of health insurance fraud in china and recommendations for fraud detection and prevention](#). *Journal of Organizational and End User Computing*, 34(4).
- Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. [API-bank: A comprehensive benchmark for tool-augmented LLMs](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, Singapore. Association for Computational Linguistics.
- Junling Liu, Peilin Zhou, Yining Hua, Dading Chong, Zhongyu Tian, Andrew Liu, Helin Wang, Chenyu You, Zhenhua Guo, LEI ZHU, and Michael Lingzhi Li. 2023. [Benchmarking large language models on cmexam - a comprehensive chinese medical exam dataset](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 52430–52452. Curran Associates, Inc.
- Vanessa Lopez, Gabriele Picco, Inge Vejsbjerg, Thanh Lam Hoang, Yufang Hou, Marco Luca Sbodio, John Segrave-Daly, Denisa Moga, Sean Swords, Miao Wei, and Eoin Carroll. 2022. [Envisioning a human-ai collaborative system to transform policies into decision models](#). *Preprint*, arXiv:2212.06882.
- Asaf Achi Mordechai, Yoav Goldberg, and Reut Tsarfaty. 2024. [NoviCode: Generating programs from natural language utterances by novices](#). *Transactions of the Association for Computational Linguistics*, 12:1330–1345.
- OpenAI. 2025. [Introducing GPT-5.2](#).
- OpenAI, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Madry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, and 400 others. 2024. [Gpt-4o system card](#). *arXiv preprint arXiv:2410.21276*.
- Zetian Ouyang, Yishuai Qiu, Linlin Wang, Gerard De Melo, Ya Zhang, Yanfeng Wang, and Liang He. 2024. [CliMedBench: A large-scale Chinese benchmark for evaluating medical large language models in clinical scenarios](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language*

- Processing*, pages 8428–8438, Miami, Florida, USA. Association for Computational Linguistics.
- Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. [The berkeley function calling leaderboard \(BFCL\): From tool use to agentic evaluation of large language models](#). In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 48371–48392. PMLR.
- Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. [Gorilla: Large language model connected with massive apis](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 126544–126565. Curran Associates, Inc.
- Sundar Pichai, Demis Hassabis, and Koray Kavukcuoglu. 2025. [A new era of intelligence with gemini 3](#).
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. [ToolLLM: Facilitating large language models to master 16000+ real-world apis](#). In *International Conference on Learning Representations*, volume 2024, pages 9695–9717.
- Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. [Abstract syntax networks for code generation and semantic parsing](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149, Vancouver, Canada. Association for Computational Linguistics.
- Marco Luca Sbodio, Vanessa López, Thanh Lam Hoang, Theodora Brisimi, Gabriele Picco, Inge Vejsbjerg, Valentina Rho, Pol Mac Aonghusa, Morten Kristiansen, and John Segrave-Daly. 2024. [Collaborative artificial intelligence system for investigation of healthcare claims compliance](#). *Scientific reports*, 14(1):11884.
- Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. [PICARD: Parsing incrementally for constrained auto-regressive decoding from language models](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Abubakr Babiker, Nathanael Schärli, Aakanksha Chowdhery, Philip Mansfield, Dina Demner-Fushman, and 13 others. 2023. [Large language models encode clinical knowledge](#). *Nature*, 620(7972):172–180.
- Zayne Sprague, Fangcong Yin, Juan Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2025. [To CoT or not to CoT? Chain-of-thought helps mainly on math and symbolic reasoning](#). In *International Conference on Learning Representations*, volume 2025, pages 94118–94162.
- Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. 2024. [MedAgents: Large language models as collaborators for zero-shot medical reasoning](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 599–621, Bangkok, Thailand. Association for Computational Linguistics.
- Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. [Large language models in medicine](#). *Nature medicine*, 29(8):1930–1940.
- Shubham Dipak Ugare, Rohan Gumaste, Tarun Suresh, Gagandeep Singh, and Sasa Misailovic. 2025. [Itergen: Iterative semantic-aware structured LLM generation with backtracking](#). In *International Conference on Learning Representations*, volume 2025, pages 24369–24394.
- Xidong Wang, Guiming Hardy Chen, Dingjie Song, Zhiyi Zhang, Zhihong Chen, Qingying Xiao, Feng Jiang, Jianquan Li, Xiang Wan, Benyou Wang, and Haizhou Li. 2024. [CMB: A comprehensive medical benchmark in Chinese](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6184–6205, Mexico City, Mexico. Association for Computational Linguistics.
- Michael Wornow, Rahul Thapa, Ethan Steinberg, Jason Fries, and Nigam Shah. 2023. [EHRSHOT: An EHR benchmark for few-shot evaluation of foundation models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 67125–67137. Curran Associates, Inc.
- Dongge Xue, Zhili Pu, Zhentao Xia, Hongli Sun, Ruihui Hou, Guangya Yu, Yupian Lin, Yongqi Fan, Jingping Liu, and Tong Ruan. 2025. [Text-to-ES bench: A comprehensive benchmark for converting natural language to Elasticsearch query](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19767–19790, Vienna, Austria. Association for Computational Linguistics.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *arXiv preprint arXiv:2505.09388*.

- Jiaxi Yang, Binyuan Hui, Min Yang, Jian Yang, Junyang Lin, and Chang Zhou. 2024. [Synthesizing text-to-SQL data from weak and strong LLMs](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7864–7875, Bangkok, Thailand. Association for Computational Linguistics.
- Guangya Yu, Yanhao Li, Zongying Jiang, Yuxiong Jin, Li Dai, Yupian Lin, Ruihui Hou, Weiyan Zhang, Yongqi Fan, Qi Ye, Jingping Liu, and Tong Ruan. 2025. [CMQCIC-bench: A Chinese benchmark for evaluating large language models in medical quality control indicator calculation](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 609–626, Vienna, Austria. Association for Computational Linguistics.
- Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. [Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.
- Hongbo Zhang, Junying Chen, Feng Jiang, Fei Yu, Zhihong Chen, Jianquan Li, Guiming Chen, Xiangbo Wu, Zhiyi Zhang, Qingying Xiao, Xiang Wan, Benyou Wang, and Haizhou Li. 2023. [HuatuogPT, towards taming language model to be a doctor](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10859–10885, Singapore. Association for Computational Linguistics.
- Xinlu Zhang, Chenxin Tian, Xianjun Yang, Lichang Chen, Zekun Li, and Linda Ruth Petzold. 2025. [Alpacare:instruction-tuned large language models for medical application](#). *Preprint*, arXiv:2310.14558.

A Supplementary Evaluation Metrics

To support the in-depth analysis, we provide formal definitions for the fine-grained metrics across rule and execution levels.

A.1 Rule-Level Analysis Metrics

These metrics decompose ORA to diagnose specific failures in logic synthesis vs. content generation.

Logic Structure Accuracy (LSA). Verifies if the Boolean operator topology (AND, OR, NOT) and nesting structure of the predicted AST are equivalent to the ground truth e , ignoring specific parameter values:

$$\text{LSA} = \mathbb{I}[\text{AST}_{\text{logic}}(\hat{e}) \equiv \text{AST}_{\text{logic}}(e)] \quad (6)$$

Function-Call Correctness (FCC). Evaluates atomic function nodes based on *Function Name Match (FNM)* and *Parameter-Schema Consistency (PSC)*:

$$\text{FCC} = \mathbb{I}[\text{FNM} = 1 \wedge \text{PSC} = 1] \quad (7)$$

PSC is satisfied ($\text{PSC} = 1$) if and only if the parameters meet four criteria: (1) Integrity, (2) Type Consistency, (3) Value Correctness, and (4) Non-redundancy.

A.2 Execution-Level Analysis Metrics

These metrics assess the robustness and efficiency of the synthesized code.

Execution Success Rate (ESR). Measures the proportion of cases where the generated code executes without runtime errors (e.g., syntax errors or invalid API calls):

$$\text{ESR} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}[\text{Exec}(\hat{f}, x_i) \text{ succeeds}] \quad (8)$$

Execution Time ($\bar{T}$). We report the average inference and execution latency per sample to evaluate efficiency (specifically used in the Paradigm Comparison).

$$\bar{T} = \frac{1}{N} \sum_{i=1}^N T_i \quad (9)$$

B Additional Experimental Analysis

B.1 Execution Success Rate Analysis

As shown in Table A1, most models achieve high Execution Success Rates (ESRs), with GPT-5.2,

Model	Duplicate	Ineligible	Excessive
qwen3-8b	88.77	43.03	83.60
qwen3-32b	96.89	92.75	99.74
qwen3-max	98.96	96.65	99.82
DeepSeek-V3.2-chat	96.03	99.07	99.91
DeepSeek-V3.2-reasoner	98.79	86.06	92.24
gpt-4o	98.27	99.26	96.38
gpt-5.2	99.83	80.39	99.12
gemini-2.5-pro	99.48	85.13	93.39
gemini-3-pro-preview	98.96	87.27	96.38
Claude-3.7-Sonnet	78.07	99.81	99.82
Claude-4.5-Sonnet	84.63	96.19	96.30
Baichuan-M2	97.93	77.51	90.56

Table A1: Execution Success Rate (ESR) across different violation types.

Gemini-2.5-Pro, and Qwen3-Max approaching or exceeding 99% in some violation types, indicating generally robust rule execution. Performance nevertheless varies across violation types: ESRs are generally high on Duplicate Charges and Excessive Charges, whereas Ineligible Drug Use exhibits lower and more variable performance across models. This suggests that the executability of generated rules remains sensitive to the characteristics of different violation types.

B.2 Reasoning-Augmented Prediction Strategy Analysis

To probe the performance ceiling of current LLMs, we evaluate three training-free, inference-time strategies beyond the zero-shot direct prompting of our main experiments: Chain-of-Thought (CoT), Self-Consistency (SC@5), and three-shot in-context learning (3-shot), which target the structured-reasoning, majority-voting, and few-shot dimensions, respectively. We study four representative models spanning open- and closed-source families and a range of parameter scales: Qwen3-8B, DeepSeek-V3.2-Chat, Gemini-3-Pro-Preview, and Claude-4.5-Sonnet. The CoT prompt asks the model to briefly reason through the rule in its own words—identifying the stated conditions, the violations they imply, and how these should be combined—before emitting the final expression. For SC@5, we sample five outputs at temperature 0.7 and take a majority vote over canonicalized expressions. For 3-shot, we prepend one demonstration per violation category (Figure A6). All other settings match the main experiments, and each configuration is evaluated on the full benchmark, with results reported in Table A2.

CoT does not consistently help and can degrade

Model	Method	Average			Duplicate Charges			Ineligible Drug Use			Excessive Charges		
		ORA	BVA	ELA	ORA	BVA	ELA	ORA	BVA	ELA	ORA	BVA	ELA
Qwen3-8B	<i>Direct</i>	26.76	72.44	43.58	9.15	60.89	29.62	21.39	70.41	30.69	49.74	86.01	70.42
	CoT	28.24	73.05	44.77	9.33	60.54	31.78	23.88	71.93	31.78	51.50	86.69	70.76
	SC@5	27.98	73.03	45.55	9.50	60.86	32.82	23.38	71.70	32.00	51.06	86.53	71.83
	3-shot	31.73	74.36	50.01	12.78	62.54	35.49	22.89	73.48	34.93	59.52	87.07	79.61
DeepSeek-V3.2-Chat	<i>Direct</i>	40.83	77.51	55.18	17.10	66.19	38.32	31.59	75.42	41.49	73.81	90.91	85.74
	CoT	41.00	77.30	56.92	17.61	65.57	39.71	29.10	75.05	41.38	76.28	91.27	89.67
	SC@5	41.47	77.81	56.89	17.27	66.97	39.64	31.84	75.47	41.71	75.31	90.99	89.31
	3-shot	42.46	78.49	57.58	18.48	66.96	39.30	32.09	76.82	42.68	76.81	91.70	90.76
Gemini-3-Pro-Preview	<i>Direct</i>	44.47	81.60	63.17	19.86	69.72	43.65	36.57	81.66	60.15	76.98	93.41	85.71
	CoT	45.89	82.18	62.64	19.69	68.74	42.66	36.07	80.40	60.74	81.92	97.39	84.52
	SC@5	45.57	81.93	63.28	20.21	69.20	42.83	37.06	82.77	60.89	79.45	93.83	86.12
	3-shot	47.81	82.44	64.32	19.86	67.65	43.01	37.06	81.72	61.90	86.51	97.96	88.05
Claude-4.5-Sonnet	<i>Direct</i>	47.02	82.22	63.22	19.34	72.04	40.31	37.06	78.94	60.22	84.66	95.67	89.13
	CoT	47.75	82.28	64.18	19.52	73.05	39.82	36.07	77.89	60.22	87.65	95.89	92.50
	SC@5	48.97	83.30	64.52	20.03	74.14	40.33	37.56	79.14	60.30	89.33	96.61	92.92
	3-shot	51.23	83.92	67.28	21.59	74.65	44.33	38.81	79.31	62.77	93.30	97.79	94.74

Table A2: Performance of representative LLMs on T2ER under different reasoning-augmented strategies. *Direct* denotes the zero-shot baseline used in the main experiments; CoT, SC@5, and 3-shot denote chain-of-thought prompting, self-consistency over five samples with majority voting, and three-shot in-context learning, respectively. ORA, BVA, and ELA denote Overall Rule Accuracy, Binary Violation Accuracy, and Execution-Label Accuracy.

strong models. While the weaker Qwen3-8B benefits modestly (+1.48 ORA), the stronger Gemini-3-Pro-Preview and Claude-4.5-Sonnet show limited or even negative effects on individual sub-tasks. This echoes prior findings that explicit step-by-step reasoning offers little benefit on deterministic, schema-grounded tasks and may induce over-generation in models that already reason implicitly (Sprague et al., 2025).

In-context learning provides the most consistent improvement. Among the three strategies, 3-shot prompting yields the largest and most stable gains (e.g., +4.97 ORA on Qwen3-8B and +4.21 on Claude-4.5-Sonnet). These improvements are mainly reflected in better output-format and structural alignment, including more accurate logical structures and function selection. However, parameter-schema consistency remains a major challenge, consistent with the dominant error pattern identified in Section 5.4.1.

Self-consistency is limited by task determinism. Since each rule admits a single correct abstract syntax tree, the variance from temperature sampling rarely aids aggregation: majority voting either recovers the original answer or amplifies a dominant error. Consequently, SC@5 delivers only modest average gains (e.g., +1.95 ORA on Claude-4.5-Sonnet) at five times the inference cost.

Inference-time enhancement yields only marginal gains. Across all four models, the three strategies improve over Direct prompting by only a

small margin in Average ORA, and none narrows the substantial gap to human performance (94.67% ORA): even the strongest configuration leaves a gap exceeding 43 points. This suggests that the bottleneck lies in parameter grounding and schema-constrained rule generation rather than inference-time prompting, consistent with the error analysis in Section 5.4.1.

C Main Error Types

In Table A3, we provide representative error cases on T2ER.

D Detailed Templates

D.1 Function bank

In Figure A1, we provide the function bank from the T2ER benchmark, including function names, descriptions, parameters, and types. The function bank covers common auditing patterns across the three major violation categories.

D.2 Electronic Medical Record Template

As shown in Figure A2, we construct a structured EMR template for downstream execution-based verification, ensuring that generated rules can be executed reliably under a unified input schema. The template covers the key modules required for insurance auditing, including basic information, examination report, laboratory report, diagnosis info,

Function Library			
function name	description	parameter	type
check_duplicate_charge	Checks whether a parent item and its child items are charged simultaneously within a scope, used to identify Duplicate Charges.	parent_item	string
		child_items	array
		scope	string
check_over_price	Performs Excessive Charges verification on billed items and checks whether the price complies with regulations.	item_name	string
		price	number
check_over_unit	Performs Excessive Charges verification on billed items and checks whether the unit complies with regulations.	item_name	string
		string	string
limitTime	Used to check whether the specific payment type and drug in the data fall within the specified time range.	drug_name	string
		payType	string
		timeLimit	string
Diagnose_Match	Detects whether the specified diagnosis name is included in the patient's diagnosis list.	drug_name	string
		targetValues	array
Inspect_Match	Detects whether the patient has evidence of the specified examination or test.	drug_name	string
		targetValues	array
Gender_Match	Determines whether the patient's gender meets the specified value.	drug_name	string
		targetValues	array
Surgery_Match	Detects whether the patient has surgical records or has undergone the specified surgery.	drug_name	string
		targetValues	array
Medicine_Match	Detects whether the patient's medication includes the specified drug name.	drug_name	string
		targetValues	array
audit_patient_symptoms	Detects whether the patient has any of the specified symptoms.	drug_name	string
		symptoms	array
valuecomparison	Performs relational operations (>, >=, <, <=, ==, !=) between the specified field and the target value.	drug_name	string
		fieldPaths	array
		operator	string
		targetValue	number

Figure A1: Overview of function bank.

medication info, surgery record and charge record, and specifies standardized fields for each module.

E Detailed Prompts

We provide the full prompts used in our experiments. The *Rule Conversion* prompt (Figure A3) elicits the text-to-executable-rule generation evaluated in the main experiments; the *Direct NL Inference* prompt (Figure A4) is used in the paradigm comparison; the *LLM-as-a-judge* prompt (Figure A5) serves as an explanation-consistency evaluator that compares human-labeled reason keywords against model- or executor-produced rationales; and the *Few-shot ICL* prompt (Figure A6) is the three-shot template used in the inference-time strategy analysis (Table A2).

Prompt for rule conversion. Figure A3 shows the prompt for converting natural-language medical insurance rules into executable rules.

Prompt for direct NL inference. Figure A4 shows the prompt for direct natural-language inference.

Prompt for LLM-as-a-judge. Figure A5 shows the LLM-as-a-judge prompt, which acts as an explanation-consistency evaluator by comparing

Electronic Medical Record		
Basic Information/ 基本信息	Patient ID	患者号
	Gender	性别
	Age	年龄
	Creation Time	创建时间
	Chief Complaint	主诉
Examination Report/ 检查报告	Examination Item	检查项目
	Examination Time	检查时间
Laboratory Report/ 检验报告	Lab Test Item	检验项目
	Test Time	检验时间
Diagnosis Info/ 诊断信息	Patient ID	患者号
	Diagnosis Name	诊断名称
Medication Info/ 用药信息	Generic Drug Name	药品通用名
	Dosage Unit	药物使用剂量单位
Surgery Record/ 手术记录	Patient ID	患者号
	Creation Time	创建时间
	Surgery Date	手术日期
	Surgery Name	手术名称
	Surgical Description	手术经过
Charge Records/ 收费报告	Patient ID	患者号
	Charge Item	收费项目名称
	Charge Date	收费日期
	Quantity	项目数量
	Unit Price	项目单价
	Unit	项目单位
	Total Cost	项目费用

Figure A2: The data schema of the Electronic Medical Record (EMR).

human-labeled reason keywords with model- or executor-produced rationales.

Prompt for few-shot in-context learning. Figure A6 shows the three-shot in-context learning prompt used in the inference-time strategy analysis (Table A2), which prepends one demonstration per violation category to the rule-conversion prompt.

Input	Ground Truth	LLM Output	Error Analysis
Drug Name: Linaclootide Capsules Note: Restricted to adults with irritable bowel syndrome with constipation (IBS-C).	valuecomparison(drug_name='Linaclootide Capsules', fieldPaths=['Medical Insurance Info:Age'], operator='>', targetValue=18) and Diagnose_Match(drug_name='Linaclootide Capsules', targetValues=['Irritable Bowel Syndrome with Constipation'])	Diagnose_Match(drug_name='Linaclootide Capsules', targetValues=['Irritable Bowel Syndrome with Constipation'])	Multi-function call error: Linaclootide Capsules are restricted to adult use, so the value comparison function should be included in the verification process, but it was omitted.
Drug Name: Hemocoagulase Bothrops Atrox Note: Restricted to surgical patients or patients with postpartum hemorrhage.	Diagnose_Match(drug_name='Hemocoagulase Bothrops Atrox', targetValues=['None']) or targetValues=['Postpartum Hemorrhage'] or not (Surgery_Match(drug_name='Hemocoagulase Bothrops Atrox', targetValues=['None']))	Surgery_Match(drug_name='Hemocoagulase Bothrops Atrox', targetValues=['None']) or Diagnose_Match(drug_name='Hemocoagulase Bothrops Atrox', targetValues=['Postpartum Hemorrhage'])	Logical connector error: Surgical records for surgical patients should not be None, so the result of the Surgery_Match function needs to be negated—i.e., prepend the not operator.
Drug Name: Compound Amino Acid (9AA) Note: Restricted to patients with renal insufficiency.	Diagnose_Match(drug_name='Compound Amino Acid (9AA)', targetValues=['Renal Insufficiency'])	audit_patient_symptoms(drug_name='Compound Amino Acid (9AA)', symptoms=['Renal Insufficiency'])	Incorrect function name selection: Renal insufficiency is a diagnostic outcome, not a clinical symptom.
Project Name: Laryngeal Tumor Resection via Direct Laryngoscopy Note: Including biopsy and pharyngeal/laryngeal foreign body removal	check_duplicate_charge(parent_item='Laryngeal Tumor Resection via Direct Laryngoscopy', child_items=['Laryngeal Biopsy via Direct Laryngoscopy', 'Pharyngeal and Laryngeal Foreign Body Removal'], scope='same_batch')	check_duplicate_charge(parent_item='Laryngeal Tumor Resection via Direct Laryngoscopy', child_items=['Biopsy', 'Pharyngeal and Laryngeal Foreign Body Removal'])	Medical terminology standardization error: The term "biopsy" generated by the large language model is overly broad and lacks proper terminology standardization.
Project Name: Total Parotidectomy Note: Including deep lobe parotid tumor resection, parotidectomy and facial nerve dissection; excluding facial nerve repair	check_duplicate_charge(parent_item='Total Parotidectomy', child_items=['Deep Lobe Parotid Tumor Resection', 'Parotidectomy', 'Facial Nerve Dissection'], scope='same_batch')	check_duplicate_charge(parent_item='Total Parotidectomy', child_items=['Deep Lobe Parotid Tumor Resection', 'Facial Nerve Dissection'])	Missing sub-items: Total parotidectomy includes three sub-procedures, but the model outputs only two.
Project Name: X-ray Computed Tomography (CT) Scan (Pelvis) (Third Body Part) Note: Billable body parts: craniocervicomedulla, chest, heart, pelvis, etc. 50% discount applies to second-hand CT equipment. Additional charge of 30 yuan for ECG or respiratory gating device. Each additional body part is charged at 60% of the previous part, with a maximum of 2 additional parts. Additional charge of 200 yuan for CTA... Unit: Per Body Part Base Price: 526.0	check_over_price(item_name='X-ray Computed Tomography (CT) Scan (Pelvis) (Third Body Part)', price=189.3) and check_over_unit(item_name='X-ray Computed Tomography (CT) Scan (Pelvis) (Third Body Part)', unit='Per Body Part')	check_over_price(item_name='X-ray Computed Tomography (CT) Scan (Pelvis) (Third Body Part)', price=526.0) and ...	Calculation error: In accordance with the rules, the price for the third body part should be calculated as base price × 60% × 60%, which equals 189.3.
Project Name: Surgical Treatment of Common Perianal Diseases (Perianal Tumor Resection) (Additional Charge for Electrocoagulation) Note: Additional charge of 50 yuan for laser, ligation, electrocoagulation, etc. Unit: Per Session. Base Price: 1344.0	check_over_price(item_name='Surgical Treatment of Common Perianal Diseases (Perianal Tumor Resection) (Additional Charge for Electrocoagulation)', price=50.0) and check_over_unit(item_name='Surgical Treatment of Common Perianal Diseases (Perianal Tumor Resection) (Additional Charge for Electrocoagulation)', unit='Per Session')	check_over_price(item_name='Perianal Tumor Resection', price=50) and check_over_unit(item_name='Perianal Tumor Resection', unit='Per Session')	Incomplete item name extraction: For example, the full item name is "Surgical Treatment of Common Perianal Diseases (Perianal Tumor Resection) (Additional Charge for Electrocoagulation)", but the large language model only outputs "Surgical Treatment of Common Perianal Diseases (Perianal Tumor Resection)".

Table A3: Error Analysis in Medical Rule Extraction.

Prompt for Medical Insurance Rules Conversion	
You are an expert Medical Insurance Policy Auditor. Your goal is to convert the Chinese medical-insurance rule texts into ONE FINAL callable expression. ### 1. Fundamental Rules - Function Selection: Use ONLY functions from the provided library. - Output Structure: Build exactly ONE result on a single line: - Either a single function call, - Or a boolean expression of function calls using and/or/not. - Parameter Format: Use NAMED PARAMETERS that EXACTLY match the function schema. ### 2. Content Extraction & Normalization Guidelines - Terminology Handling: - Keep Chinese terms unchanged. - Extract the input item name without making any changes; never omit content inside parentheses. - Extract standard medical terminology from the input fields, such as child_item names and diagnosis names. - Normalization: Do NOT simply copy the description. Map description text to standard Medical Service Project Names. - Calculation Logic: - Numbers are numeric literals. - Extract price calculation rules from the input fields and compute the final price; note that for additional charges, output ONLY the amount of the additional charge, not the total. - Completeness: - A rule typically contains multiple checks, such as "price" and "unit". Must call the complete checklist. ### 3. Output Format & Exceptions - Formatting: Output STRICTLY the expression line. Do NOT use Markdown code fences (``), JSON, or explanations. --- FUNCTION LIBRARY (JSON Raw): <pre>{raw}</pre> ### Instruction Analyze the user input and generate the callable expression based on the rules above.	您是一位专业的医疗保险政策审核专家。您的目标是将中文医保规则文本转换为**一个最终可调用的表达式**。 ### 1. 基本规则 - 函数选择: 仅使用所提供的函数库中的函数。 - 范围: 仅针对规则中明确规定的项目生成检查逻辑，除非明确要求。 - 输出结构: 严格在单行内构建一个结果: - 可以是一个单独的函数调用, - 或者是一个使用 and/or/not 连接函数调用的布尔表达式。 - 参数格式: 使用命名参数, 且必须完全匹配的函数模式定义。 ### 2. 内容提取与标准化准则 - 术语处理: - 保持中文术语不变。 - 提取输入项目名称, 不作任何修改; 切勿省略括号内的内容。 - 从输入字段 (如项目名称、诊断名称等) 中提取标准的医学术语。 - 标准化: 不要 简单复制描述文本。需将描述文本**映射到** 标准的"医疗服务项目名称"。 - 计算逻辑: - 数字使用数值字面量。 - 从输入字段中提取价格计算规则并计算最终价格; 请注意, 对于附加费用 (如 "加收"), 仅输出附加费用的金额, 而非总金额。 - 完整性: - 一条规则通常包含多个检查项 (如 "价格" 和 "单位")。必须调用完整的检查列表。 ### 3. 输出格式与例外 - 格式化: 严格输出表达式行。不要使用 Markdown 代码块 (``)、JSON 或任何解释性文字。 - 禁止项: 不要输出任何 `<think>` 标签或内部推理过程。只输出最终结果。</think> --- 函数库 (JSON 原始格式): <pre>{raw}</pre> ### 指令 请分析用户输入, 并根据以上规则生成可调用的表达式。

Figure A3: Prompt for Medical Insurance Rules Conversion.

Prompt for Direct Natural Language Inference	
# Role You are a professional Medical Insurance Compliance Auditor. Your core competency involves performing logical assessments of patient medical records or billing details (Patient Cases) based on provided natural language medical insurance rules (Policies), and outputting precise reasons for any violations. # Task The input data consists of: content: A medical insurance policy described in natural language (including item definitions, pricing standards, restrictive conditions, etc.). patient_cases: A set of structured patient cases and billing details. You are required to evaluate whether each patient_case violates the policy and output the results in JSON format, strictly adhering to the [Reasoning Generation Logic]. # Instruction: Refer to the following medical insurance violation scenarios: Duplicate Charges: The policy implies an inclusion relationship between medical items. It is a violation if a parent item and any of its child items are charged simultaneously. Excessive Charges: It is a violation if the price or unit of a charged item does not conform to the standard price or unit. Ineligible Drug Use: It is a violation if a restricted drug is prescribed or administered without satisfying the policy's preconditions. # Examples (few-shots)	# 角色 你是一名专业的医保违规智能稽核员。你的核心能力是基于给定的自然语言医保规则, 对患者的病历或收费明细进行逻辑判断, 并输出精确的违规原因。 # 任务 输入数据包含: 1. `content`: 一条自然语言描述的医保规则 (含项目内涵、价格标准、限定条件等)。 2. `patient_cases`: 一组结构化的患者病例及收费明细。 你需要逐一判断每个 patient_case 是否违反规则, 并严格按照【归因生成逻辑】输出 JSON 结果。 # 指令: 参考以下这些医保违规情况: - 重复收费: 规则中隐含医疗项目间的包含关系, 父项目与其任一子项目在同一时间收费即违规。 - 超标准收费: 收费项目的价格和单位不符合标准价格和单位即违规。 - 用药限制: 受限药品在未满足规则前置条件的情况下被开具/使用即违规。 # Examples (few-shots)

Figure A4: Prompt for Direct Natural Language Inference.

Prompt for LLM-as-a-judge	
# Role You are a senior Medical Insurance Risk Management Expert and Logic Validator. Your core competency is the ability to precisely identify subtle nuances in medical rule descriptions—including numerical values, clinical diagnosis scopes, drug indications, and billing logic—and to decompose natural language descriptions into independent logical units for comparison and scoring. # Task You need to compare the human-labeled expected_reason_keyword with the algorithm-generated real_reason_keyword to determine if they are completely consistent in terms of business logic. You need to decompose the text into independent atomic violation points, judge how much of the expected content is covered by the real content, and calculate a consistency score (0-100). # Evaluation Guidelines (Core Judgment Criteria) ## Step 1: Logical Decomposition First, decompose the expected_reason_keyword into N independent "atomic violation points" based on logical connectors (e.g., "and", "but", "also", " ", ":", ";"). * Example: "A violation but B violation" -> Decomposed into [A, B], total N=2 points. * Example: "A violation" -> Decomposed into [A], total N=1 point. ## Step 2: Item-by-Item Matching Check if the real_reason_keyword contains each of the aforementioned atomic points. Matching Standards (if a point fails any of these, it is considered a Mismatch): 1. **Values/Units**: Numbers must match exactly (e.g., "7 yuan" vs "25 yuan" is a mismatch); unit semantics must be consistent ("3 days" vs "3 days (different phrasing)" is a match). - Check specific numbers in the description (e.g., amount, days, frequency, dosage). - Rule: If expected is "exceeded cost by 7", and real is "exceeded cost by 25", it is considered a mismatch. - Rule: If expected is "3 days quantity", and real is "3 days amount", and the unit semantics are the same, it is considered a match. 2. **Clinical Scope**: Diagnosis/Symptoms/Drugs must be consistent (e.g., "Gastric Ulcer" vs "Gastric Ulcer with Hemorrhage" have different scopes and are considered a mismatch). - Strictly distinguish between "Diagnosis" and "Symptom". - Strictly distinguish between "Underlying Disease" and "Comorbidities/Complications". - Rule: If expected is "does not have Diagnosis A", and real is "does not have Diagnosis A and accompanied by Symptom B", the scopes are inconsistent, considered a mismatch. - Rule: If expected is "Child", and real is "Under 14 years old", semantics are equivalent, considered a match. 3. **Core Entities**: Nursing levels/Project names must be consistent. - Rule: e.g., "Strict Isolation Nursing" vs "Level I Nursing", entities are different, considered a mismatch. # Examples (Few-Shot)	# 角色 你是一位资深的医疗保险风控专家及逻辑校验员。你的核心能力是能够精确识别医疗规则描述中的细微差异，包括数值、临床诊断范围、药品适应症以及收费逻辑，并将自然语言描述拆解为独立的逻辑单元进行比对打分。 # 任务 你需要对比人工标注的预期原因（expected_reason_keyword）和算法生成的实际原因（real_reason_keyword），判断它们在业务逻辑上是否**完全一致**。你需要将文本拆解为独立的**违规原子项**，判断real`覆盖了多少`expected`的内容，并计算一致性得分（0-100）。</p> <p><b># 评估指导：核心判断准则</b> ## 步骤 1：逻辑拆解 首先，根据逻辑连接词（如 "and", "but", "且", " ", ":", ";"）将`expected_reason_keyword`拆解为 N 个独立的“原子违规点”。 * 示例: "A违规 but B违规" -> 拆解为 [A, B], 共 N=2 个点。 * 示例: "A违规" -> 拆解为 [A], 共 N=1 个点。 ## 步骤 2：逐项匹配 检查`real_reason_keyword`中是否包含了上述每一个原子点。 匹配标准如下（只要有一条不满足，该原子点视为**不匹配**）： 1. **数值/单位**: 数字必须精确匹配（如"7元" vs "25元"为不匹配），单位语义需一致（"3日" vs "3天"为匹配）。 - 检查描述中的具体数字（如金额、天数、次数、用量）。 - 规则：如果预期是"超标7费用"，实际是"超标25费用"，视为不匹配。 - 规则：如果预期是"3日量"，实际是"3天量"，单位语义相同，视为匹配。 2. **临床范围**: 诊断/症状/药品必须一致（如"胃溃疡" vs "胃溃疡伴出血"范围不同，视为不匹配）。 - 严格区分"诊断"与"症状"。 - 严格区分"基础疾病"与"合并症/并发症"。 - 规则：如果预期是"不具备A诊断"，实际是"不具备A诊断且伴随B症状"，范围不一致，视为不匹配。 - 规则：如果预期是"儿童"，实际是"14岁以下"，语义对等，视为匹配。 3. **核心实体**: 护理等级/项目名称必须一致。 - 规则：如"严密隔离护理" vs "一级护理"，实体不同，视为不匹配。 # Examples (Few-Shots)

Figure A5: Prompt for LLM-as-a-judge.

Prompt for few-shot in-context learning	
You are an expert Medical Insurance Policy Auditor. Your goal is to convert the Chinese medical-insurance rule texts into ONE FINAL callable expression. ### 1. Fundamental Rules - Function Selection: Use ONLY functions from the provided library. - Output Structure: Build exactly ONE result on a single line: - Either a single function call. - Or a boolean expression of function calls using and/or/not. - Parameter Format: Use NAMED PARAMETERS that EXACTLY match the function schema. ### 2. Content Extraction & Normalization Guidelines - Terminology Handling: - Keep Chinese terms unchanged. - Extract the input item name without making any changes; never omit content inside parentheses. - Extract standard medical terminology from the input fields, such as child_item names and diagnosis names. - Normalization: Do NOT simply copy the description. Map description text to standard Medical Service Project Names. - Calculation Logic: - Numbers are numeric literals. - Extract price calculation rules from the input fields and compute the final price; note that for additional charges, output ONLY the amount of the additional charge, not the total. - Completeness: - A rule typically contains multiple checks, such as "price" and "unit". Must call the complete checklist. ### 3. Output Format & Exceptions - Formatting: Output STRICTLY the expression line. Do NOT use Markdown code fences (``), JSON, or explanations. --- FUNCTION LIBRARY (JSON Raw): <pre>{raw}</pre> Example 1: Input: Item Name: Renal Tubular Phosphate Reabsorption Test. Item Description: Includes fixed calcium-phosphorus diet, distilled water intake, urine volume measurement at 1 and 2 hours after water intake for two consecutive days, blood and urine creatinine tests, calcium and phosphorus tests, and result analysis. Output: <pre>check_duplicate_charge(parent_item="Renal Tubular Phosphate Reabsorption Test", child_items=["Creatinine Test", "Calcium Test", "Inorganic Phosphorus Test"], scope="same_batch")</pre> Example 2: Input: "Drug Name": "Ornithine Aspartate", "Note": "Restricted to patients with hyperammonemia or hepatic encephalopathy." Output: <pre>Diagnose_Match(drug_name="Ornithine Aspartate", targetValues=["Hepatic Encephalopathy"]) or audit_patient_symptoms(drug_name="Ornithine Aspartate", symptoms=["Hyperammonemia"])</pre> Example 3: Input: "Item Name": "Activities of Daily Living Assessment Scale (50% surcharge for computer-based self-assessment)", "Description": "Assessment duration within 30 minutes; computer-based self-assessment is charged an additional 50% based on the corresponding item.", "Unit": "per session", "Base Price": 33.0 Output: <pre>check_over_price(item_name="Activities of Daily Living Assessment Scale (50% surcharge for computer-based self-assessment)", price=16.5) and check_over_unit(item_name="Activities of Daily Living Assessment Scale (50% surcharge for computer-based self-assessment)", unit="per session")</pre> ### Instruction Analyze the user input and generate the callable expression based on the rules above.	您是一位专业的医疗保险政策审核专家。您的目标是将中文医保规则文本转换为 一个最终可调用的表达式。 ### 1. 基本规则 - 函数选择: 仅使用所提供的函数库中的函数。 - 范围: 仅针对规则中明确规定的项目生成检查逻辑。 - 输出结构: 严格在单行内构建一个结果: - 可以是一个单独的函数调用, - 或者是一个使用 and/or/not 连接函数调用的布尔表达式。 - 参数格式: 使用命名参数, 且必须完全匹配的函数模式定义。 ### 2. 内容提取与标准化准则 - 术语处理: - 保持中文术语不变。 - 提取输入项目名称, 不作任何修改; 切勿省略括号内的内容。 - 从输入字段 (如项目名称、诊断名称等) 中提取标准的医学术语。 - 标准化: 不要 简单复制描述文本。需将描述文本映射到 标准的“医疗服务项目名称”。 - 计算逻辑: - 数字使用数值字面量。 - 从输入字段中提取价格计算规则并计算最终价格; 请注意, 对于附加费用 (如“加收”), 仅输出附加费用的金额, 而非总金额。 - 完整性: - 一条规则通常包含多个检查项 (如“价格”和“单位”)。必须调用完整的检查列表。 ### 3. 输出格式与例外 - 格式化: 严格输出表达式行。不要使用 Markdown 代码块 (``)、JSON 或任何解释性文字。 - 禁止项: 不要输出任何 `<think>` 标签或内部推理过程。只输出最终结果。</think> 函数库 (JSON 原始格式): <pre>{raw}</pre> ### 示例 示例 1: 输入: 项目名称: 肾小管磷重吸收试验 项目内涵: 含固定钙磷饮食、双蒸水饮用, 连续两日饮水后1、2小时测尿量, 查血尿酸和钙磷及结果分析 输出: <pre>check_duplicate_charge(parent_item="肾小管磷重吸收试验", child_items=["肌酐测定", "钙测定", "无机磷测定"], scope="same_batch")</pre> 示例 2: 输入: 药品名称: "门冬氨酸鸟氨酸", 备注: "限血氨升高或肝性脑病的患者。" 输出: <pre>Diagnose_Match(drug_name='门冬氨酸鸟氨酸', targetValues=['肝性脑病']) or audit_patient_symptoms(drug_name='门冬氨酸鸟氨酸', symptoms=['血氨升高'])</pre> 示例 3: 输入: 项目名称: "日常生活能力评定量表(使用电脑自测的量表加收50%)", 说明: "测量时间30分钟以内, 电脑自测的量表按相应项目加收50%", 单位: "次", 基准价格: 33.0 输出: <pre>check_over_price(item_name="日常生活能力评定量表(使用电脑自测的量表加收50%)", price=16.5) and check_over_unit(item_name="日常生活能力评定量表(使用电脑自测的量表加收 50%)", unit="次")</pre> ### 指令 请分析用户输入, 并根据以上规则生成可调用的表达式。

Figure A6: Prompt for Few-shot In-Context Learning.