A Hybrid Framework for Financial Regulatory Compliance: Integrating LLMs and SMT Solvers for Automated Legal Analysis

Yung-Shen Hsia
Department of Management Information Systems,
National ChengChi University
Taipei, Taiwan
113356046@g.nccu.edu.tw

Fang Yu
Department of Management Information Systems,
National ChengChi University
Taipei, Taiwan
yuf@nccu.edu.tw

Abstract

This paper presents a hybrid framework that combines Large Language Models (LLMs) with Satisfiability Modulo Theories (SMT) solvers to advance automated legal analysis in the financial regulatory domain. Designed for Taiwan's legal context, the system integrates a Retrieval-Augmented Generation (RAG) pipeline to access statutory references, historical enforcement cases, and corporate violation records from the Financial Supervisory Commission (FSC). Leveraging Python-based tools including Z3 and OpenAI API, the framework supports case similarity assessment, regulatory constraint checking, and scenario simulation through constraint-based formal reasoning.

The agent emulates legal expert analysis using a chain-of-thought prompting approach tailored for financial compliance. It identifies potential violations, interprets applicable legal provisions, and generates corrective recommendations aligned with statutory requirements. By distinguishing hard constraints (laws) from soft constraints (case-specific facts), the system enables minimal legal corrections and iterative validation using a cone-of-influence analysis. A Gradio-based interactive interface allows users to explore compliance evaluations in real-time.

Experimental evaluation on 87 FSC-issued penalty cases shows that the proposed LLM+SMT hybrid system significantly outperforms LLM-only baselines in identifying illegal terms, generating accurate corrections, and delivering interpretable reasoning outputs—achieving over 300× speed-up and perfect accuracy in constraint satisfaction tasks. This work lays the groundwork for future applications of AI-enhanced legal reasoning in financial governance, offering a transparent, customizable, and regulation-aligned tool for automated compliance diagnostics.

CCS Concepts

• **Theory of computation → Constraint and logic programming**; **Verification by model checking**; *Linear logic*.

KDD-MLF, Toronto, Canada

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/10.1145/XXXXXXX.XXXXXXX

Keywords

Financial precedent analysis, Retrieval-Augmented Generation (RAG), SMT Solver, legal tech, legal understanding analysis, constraint solving, Financial Supervisory Commission (FSC).

ACM Reference Format:
Yung-Shen Hsia and Fang Yu. 2025. A Hybrid Framework for Financial Regulatory Compliance: Integrating LLMs and SMT Solvers for Automated Legal Analysis. In *Proceedings of KDD Workshop on Machine Learning in Finance (KDD-MLF).* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/XXXXXXX.XXXXXXX

1 Introduction

As legal regulations grow more intricate and frequently revised, automating compliance has become a pressing challenge in Legal Tech. A key difficulty lies in translating legal provisions into machine-readable formats while preserving their semantic nuance and contextual dependencies [15, 18]. Viewing laws as "computable" helps reduce human error and labor costs by enabling structured, rule-based interpretation [21].

Artificial intelligence (AI) has become integral to the digital transformation of financial institutions, notably through machine learning, process automation, predictive analytics, and chatbots [16]. In regulatory and compliance contexts, AI-driven anomaly detection and statistical modelling have been widely applied to monitor suspicious transactions, detect money laundering, and automate reporting workflows [7, 11, 17]. These technologies not only improve detection accuracy and processing speed but also allow compliance teams to focus on high-value investigations [2].

Early work in software verification introduced techniques such as model checking and SAT/SMT solving to ensure correctness under constraints [4, 9, 10]. Recent studies extend these methods to regulatory compliance, framing laws and case facts as formal constraints to assess risk and support governance [8].

Large Language Models (LLMs) have shown strong performance in language understanding and generation, making them suitable for interpreting legal texts and enforcement cases [3, 20]. Yet, their free-form outputs often lack the precision needed for regulatory compliance. Augmenting LLMs with structured retrieval or validation modules—e.g., through Retrieval-Augmented Generation (RAG)—can improve their factual consistency and analytical rigor [14].

This study introduces a hybrid compliance analysis agent that combines SMT Solvers with LLMs, tailored for Taiwan's financial legal context. Built on a RAG pipeline, the system integrates penalty case archives, statutory references, and violation records issued by the Financial Supervisory Commission (FSC). It uses Python

tools—such as Z3 and NetworkX—for constraint solving, case similarity matching, and scenario simulation.

By encoding legal texts as constraints and embedding explainable reasoning, our system addresses current limitations in compliance automation caused by the diversity and complexity of legal sources. The integration of generative and formal methods allows for more precise, traceable, and regulation-aligned outputs in high-stakes financial contexts. This approach enhances regulator–institution communication and offers a foundation for scalable Legal Tech applications.

We propose an integrated regulatory analysis framework that combines large language models (LLMs) with a Satisfiability Modulo Theories (SMT) solver. This approach addresses the inherent reasoning limitations of current LLMs by enabling more accurate identification and interpretation of complex legal violations. The framework allows customization of parameters according to user needs, facilitating the prediction of potential violations and suggesting preventive measures.

To evaluate our proposed method, we conducted a comprehensive analysis of 89 real-world enforcement cases issued by Taiwan's Financial Supervisory Commission. Our framework successfully identified illegal contractual terms and recommended pertinent regulatory improvements. The inclusion of the SMT solver significantly enhanced our ability to reason about and detect complex violations, thereby improving both the accuracy and interpretability of the analysis.

2 Related Work

Numerous studies have explored the automation of regulatory compliance through various computational techniques.

Beach et al. [5] proposed a semantic framework that decouples legal domain expertise from system implementation. This separation empowers non-technical experts to participate in compliance modeling, advancing the accessibility of legal automation.

In financial auditing, Berger et al. [6] evaluated Large Language Models (LLMs) for automated checks using ALI and ZeroShotALI systems. Comparing models such as LLaMA-2 and GPT-4 against IFRS and HGB-aligned datasets, they found GPT-4 excelled in multilingual settings, while LLaMA-2 offered cost-effective performance for non-compliance detection. However, model variability and false positives remain challenges. Fine-tuning on domain-specific corpora is recommended to address performance, privacy, and cost trade-offs.

Beyond LLMs, SMT solvers have been leveraged for legal reasoning. Judson et al. [13] introduced *soid*, a system combining Z3 and symbolic execution (e.g., KLEE) to support formal accountability via a GUI, facilitating access for legal practitioners.

Feng et al. [12] translated legal constraints into SMT via Metric First-Order Temporal Logic (MFOTL), refining results through counterexamples. Their Python-based implementation enables efficient compliance checking across healthcare and finance.

Tsigkanos et al. [22] applied SMT to model economic transaction networks, using logical constraints to ensure macro–micro consistency and sectoral balance.

Separately, research on counterfactual explanations [23, 1] highlights their utility in making algorithmic decisions transparent and contestable, particularly under data protection regulations like GDPR.

The Semantic Self-Verification (SSV) framework [19] integrates LLMs with SMT solvers: the LLM drafts formal constraints, which are verified and refined based on solver feedback. This closed loop significantly improves accuracy in tasks like ProofWriter and PrOntoQA, minimizing manual formalization.

Building on these directions, our work applies SMT–LLM integration to regulatory compliance. By incorporating counterfactual analysis and constraint solving into a unified pipeline, we address key challenges such as multilingual applicability, non-technical accessibility, and the need for interpretable and adaptive compliance feedback. Our approach demonstrates how combining the rigor of SMT with the generative capabilities of LLMs can produce trustworthy and context-aware compliance automation.

3 Methodology

To provide a comprehensive overview of our compliance framework, Figure 1 illustrates the *Interactive Compliance Pipeline*, which consists of two interconnected modules: a fully automated compliance evaluation engine and an interactive constraint-exploration interface. These modules work in concert to assess legal scenarios, detect violations, and suggest corrective measures, all grounded in symbolic reasoning and generative language models.

3.1 System Architecture Overview

Figure 1 shows how a user-submitted query is processed through the system:

- The query is rewritten and decomposed using LLM-based templates.
- Relevant laws and precedents are retrieved via a hybrid RAG search engine.
- Legal texts are transformed into SMT constraints using a structured prompting pipeline.
- The SMT solver evaluates compliance and offers suggestions for resolution.
- An interactive module enables users to iteratively modify assumptions and explore legally compliant alternatives.

This architecture supports both automatic and user-guided analysis, enabling transparent, adaptive legal reasoning.

3.2 Data Collection and Preprocessing

As shown in Figure 2, we collected 480 legal case summaries from the official website of Taiwan's Financial Supervisory Commission (FSC) via dynamic web scraping. Selenium[1] was used to automate interactions with the FSC's case announcement system, enabling the extraction of content rendered through dropdowns and JavaScript.

The dataset includes metadata and full-text descriptions of administrative and financial cases (see Appendix for data format examples). We then applied a Large Language Model (LLM) to identify legal references cited within the narratives, typically in the form of article-based citations from applicable laws.

[1] https://www.fsc.gov.tw/ch/home.jsp?id=131&parentpath=0,2

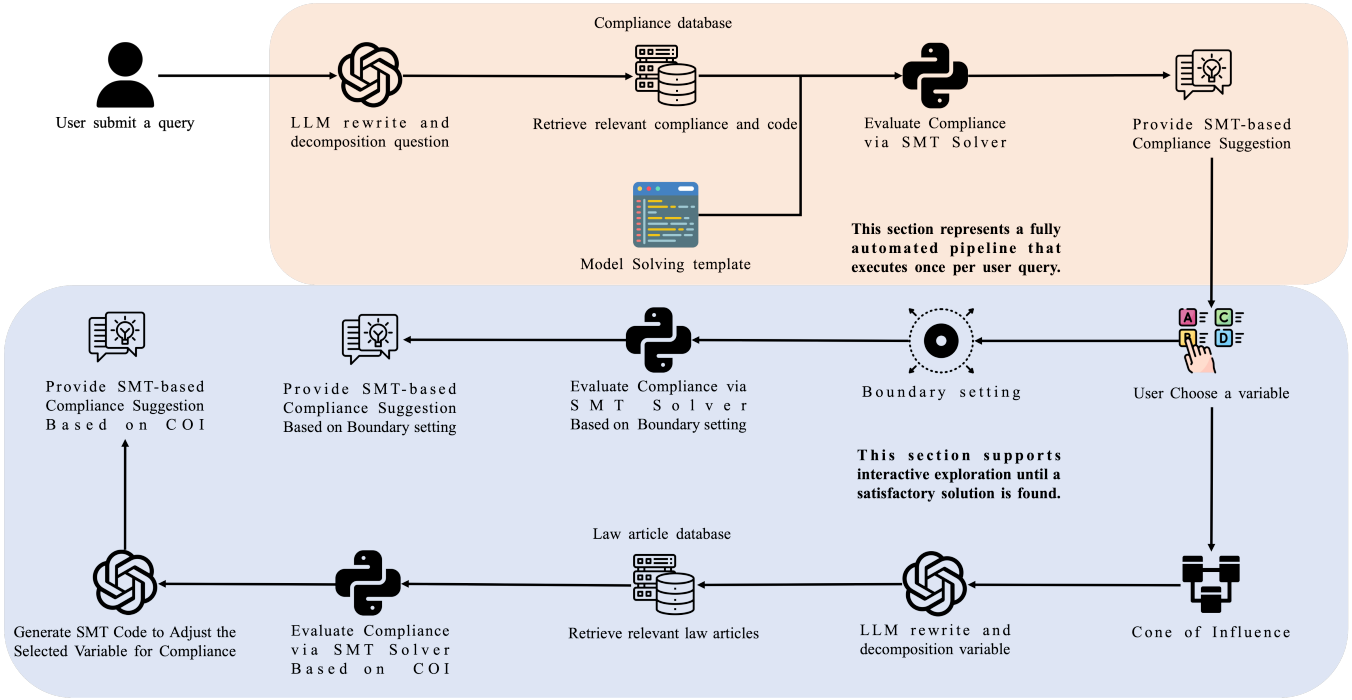

Figure 1: Interactive Compliance Pipeline

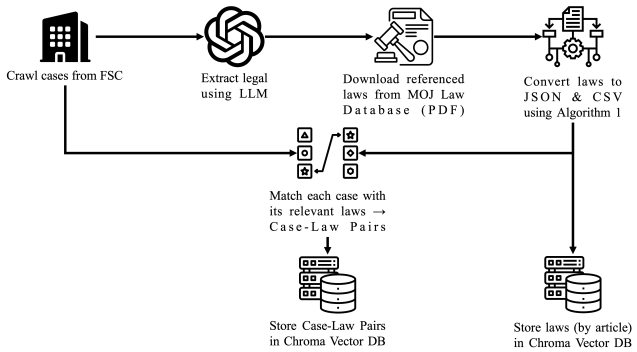

Figure 2: System Implementation

To obtain the full text of the referenced laws, we queried Taiwan's official Law Database[2] and downloaded the corresponding PDF files. Legal articles and clauses were extracted using a rule-based parser (Algorithm 1), producing structured JSON and CSV representations.

Each case was paired with its relevant legal provisions, establishing case–law mappings. These provisions were then indexed in ChromaDB at the article level, supporting downstream legal retrieval and case-based reasoning via semantic search.

3.3 Build SMT Hard Constraints Based on Laws

Legal reasoning requires that all actions comply with statutory norms. We adopt an SMT solver to verify whether a proposed action satisfies such legal constraints. Each statute is encoded as a set of Boolean formulas—typically conjunctions or implications—and any violation is detected as an unsatisfiable condition. Furthermore, the solver identifies which constraints fail, enhancing interpretability for downstream analysis.

Consider Article 143-6 of Taiwan's Insurance Act, which mandates escalating supervisory measures as capital adequacy declines from "under-capitalized" to "significantly" and "critically undercapitalized." Corresponding actions range from remediation plans to license revocation (see Appendix A).

We encode this logic using propositional formulas:

$$CapOK = \neg UnderCap \wedge \neg SevUnderCap \wedge \neg CritUnderCap,$$
$$AnyMeas = Meas_1 \vee Meas_2 \vee \cdots \vee Meas_N.$$

$$CapOK \rightarrow \neg AnyMeas \tag{1}$$
$$UnderCap \wedge \neg Plan \rightarrow AnyMeas \tag{2}$$
$$UnderCap \wedge Plan \rightarrow \text{true} \tag{3}$$
$$SevUnderCap \rightarrow (AnyMeas \wedge Dismiss) \tag{4}$$
$$CritUnderCap \rightarrow (AnyMeas \wedge Discipline) \tag{5}$$

Here, *UnderCap*, *SevUnderCap*, and *CritUnderCap* denote levels of under-capitalization; *Plan* indicates the presence of a remediation plan; $Meas_i$ represents individual supervisory measures; and *Dismiss*, *Discipline* refer to executive actions and penalties under Article 149-3-1.

If capital is deemed adequate (*CapOK*), then no measures should be taken (1). When capital is inadequate and no remediation plan

[2]http://law.moj.gov.tw

exists, some supervisory measure must be enforced (2); otherwise, compliance is violated.

As an example, suppose an insurance firm is undercapitalized without submitting a plan:

$$\mathit{UnderCap} = \text{true}, \quad \mathit{Plan} = \text{false}, \quad \mathit{AnyMeas} = \text{false}$$

This contradicts implication (2), resulting in an unsatisfiable constraint:

$$\mathit{UnderCap} \wedge \neg \mathit{Plan} \wedge \neg \mathit{AnyMeas}.$$

This conflict flags the case as non-compliant. The logical formulation enables systematic detection of legal violations through SMT-based verification, providing a robust framework for automated legal compliance checking.

3.4 Build SMT Soft Constraints for Legal Compliance

To improve legal interpretability, we transform scenario-specific facts into *soft constraints*, while retaining statutory provisions as *hard constraints*. This MaxSMT formulation allows the solver to suggest minimal corrections—i.e., flipping as few Booleans as possible—to restore compliance in unsatisfiable scenarios.

For instance, consider the hard constraint:

$$\mathit{UnderCap} \wedge \neg \mathit{Plan} \rightarrow \mathit{AnyMeas}$$

and the soft scenario:

$$\mathit{UnderCap} = \text{true}, \quad \mathit{Plan} = \text{false}, \quad \mathit{AnyMeas} = \text{false}$$

which leads to a violation. The solver can return minimal corrections:

Case 1: Flip *Plan* = true, satisfying:

$$\mathit{UnderCap} \wedge \mathit{Plan} \rightarrow \text{true}$$

Case 2: Flip *AnyMeas* = true, satisfying:

$$\mathit{UnderCap} \wedge \neg \mathit{Plan} \rightarrow \mathit{AnyMeas}$$

Both incur a soft cost of 1. This method mirrors real-world legal reasoning, emphasizing not just violation detection but also actionable remediation.

3.5 Supplementary regulations via Retrieval-Augmented-Generation

The Supplementary regulations Retrieval-Augmented-Generation algorithm (Algorithm 2) is designed to expand an original legal clause L_0 by extracting pertinent supplementary clauses from a comprehensive legal database $\mathcal{D}$. This process comprises four principal stages:

(1) **Query Generation:** An LLM is invoked to generate a set of semantically diverse queries $\mathcal{Q} = \{q_1, \ldots, q_m\}$ based on the input clause L_0. These queries capture different facets of the legal context embedded in L_0.

(2) **Hybrid Retrieval:** For each query $q \in \mathcal{Q}$, the system performs two parallel retrievals on $\mathcal{D}$:
- A lexical match via BM25, yielding $R_{\text{bm25}} = \text{BM25Search}(\mathcal{D}, q, K)$.
- A semantic match via vector embeddings, yielding $R_{\text{vec}} = \text{VectorSearch}(\mathcal{D}, q, K)$.

Algorithm 1: Article and Clause Extraction from PDF

Input: PDF document D
Output: Structured dictionary A containing articles and clauses

```
Initialize empty string T ← "";
Open PDF file D using pdfplumber;
foreach page p in D do
    t ← extract_text(p);
    if t ≠ None then
        T ← T‖t;
Split T into lines L;
Initialize empty dictionary A, current article ID a ← None,
  current clause c ← None;
foreach line l in L do
    Clean l by stripping and collapsing multiple whitespaces
      into a single space;
    if l matches regex pattern for "section headings" then
        continue;
    if l matches pattern "Article n" then
        a ← n;
        Initialize A[a] with fields title: l, empty list
          clauses, and empty content;
        c ← None;
    else if a ≠ None then
        if l matches clause pattern "m. text" or "m, text"
          then
            Append text to A[a][clauses];
            c ← text;
        else
            if c ≠ None then
                Concatenate l to the last clause in
                  A[a][clauses], removing spaces;
            else
                Concatenate l to A[a][content], removing
                  spaces;
return A;
```

The results are combined into a unified set $R = R_{\text{bm25}} \cup R_{\text{vec}}$. Each document $d \in R$ is then assigned a *hybrid score*:

$$s_{\text{hybrid}}(d, q) = \alpha \, \text{sim}_{\text{vec}}(d, q) + (1 - \alpha) \, \text{score}_{\text{bm25}}(d, q),$$

$$\alpha \in [0, 1].$$

(3) **Reranking and Filtering:** The top-K candidates by hybrid score are re-ranked using a supervised Cross-Encoder model, implemented as the `FlagReranker`. The refined list R' is then passed through an LLM-based filter `LLM_FilterUseful`, which selects only those clauses that are both contextually relevant and non-redundant with respect to L_0.

(4) **Aggregation:** All filtered clauses C from each query are aggregated into a global set $\mathcal{S}$. Finally, duplicates are removed to yield the unique supplementary clause set.

We encapsulate the retrieval pipeline into a Python module named `LegalSearchEngine`, which integrates vector-based retrieval, lexical indexing, hybrid scoring, reranking, and clause classification. Detailed architecture, prompt configurations, and implementation parameters of `LegalSearchEngine` are provided in Appendix D.

Following this procedure, we obtain a structured financial case instance paired with relevant legal clauses. These results are subsequently translated into SMT constraints for downstream processing.

Algorithm 2: Supplementary regulations via Retrieval-Augmented-Generation

Input: Original clause L_0; legal database $\mathcal{D}$; hybrid weight $\alpha \in [0,1]$; retrieval size K; rerank size N
Output: Supplementary clauses set $\mathcal{S}$
$\mathcal{Q} \leftarrow \text{LLM\_GenQueries}(L_0)$;
$\mathcal{S} \leftarrow \emptyset$;
foreach $q \in \mathcal{Q}$ **do**
  $R_{\text{bm25}} \leftarrow \text{BM25Search}(\mathcal{D}, q, K)$;
  $R_{\text{vec}} \leftarrow \text{VectorSearch}(\mathcal{D}, q, K)$;
  $R \leftarrow R_{\text{bm25}} \cup R_{\text{vec}}$;
  foreach $d \in R$ **do**
    compute
    $$s_{\text{hybrid}}(d) = \alpha\, \text{sim}_{\text{vec}}(d,q) + (1-\alpha)\, \text{score}_{\text{bm25}}(d,q)$$
    ;
  $R_{\text{top}} \leftarrow \text{TopK}(R, s_{\text{hybrid}}, K)$;
  $R' \leftarrow \text{Rerank}(R_{\text{top}}, \text{CrossEncoder}, q)$;
  $C \leftarrow \text{LLM\_FilterUseful}(R', L_0)$;
  $\mathcal{S} \leftarrow \mathcal{S} \cup C$;
return $\text{Unique}(\mathcal{S})$;

3.6 SMT Constraints Generation

We employ OpenAI's API to generate SMT code, guided by five key principles for legal soundness and interpretability:

- **Few-shot examples**: We prompt the LLM with a well-formed constraint sample to guide syntactic correctness.
- **Typed declarations**: Variables are declared with appropriate Z3 types (e.g., `Real()`, `Bool()`).
- **Constraint separation**: Statutory rules are hard constraints; scenario facts are soft constraints for flexible inference.
- **Bound control**: Numeric variables are bounded to avoid invalid or extreme solutions.
- **Result display**: Solutions are reported with case comparisons for validation and traceability.

This structured approach ensures that the generated SMT models are both compliant and informative, aligning legal logic with scenario-specific details.

3.7 Directed Inference Core

The Cone of Influence (COI) is a classic concept in formal verification, used to reduce the search space of a model by focusing only on variables and constraints that can affect a particular output or "root variable." By isolating the subset of logic that influences a specific compliance outcome, we can minimize unnecessary computation and enhance explainability. In our setting, COI is used to trace how changing a particular variable (e.g., a firm's capital grade) impacts other interconnected constraints and financial indicators. This makes the logic path more interpretable and facilitates localized legal adjustments.

We implement a *Directed Inference Core* based on the COI principle to trace how a target variable's value can be flipped and what dependencies must adjust to maintain compliance. As detailed in

Algorithm 3: Directed Inference Core

Input: Optimize instance *opt*, root variable *root_var*
Output: Core constraints set *core_cons*, Core variable domains *core_var_domains*
Initialize *varDepGraph* ← {}; Initialize *varSorts* ← {};
Initialize *eqCons* ← [], *otherCons* ← [];
foreach *constraint c in opt.assertions()* **do**
  foreach *variable v in getVars(c)* **do**
    varSorts[*v*.name] ← *v*.sort();
  if *isEq(c)* **then**
    (*lhs*, *rhs*) ← *c*.children();
    L ← getVars(*lhs*), *R* ← getVars(*rhs*);
    foreach *lv in L* **do**
      ln ← *lv*.name;
      if *R* = ∅ **then**
        varDepGraph[*ln*].add(*ln*);
      foreach *rv in R* **do**
        rn ← *rv*.name;
        varDepGraph[*ln*].add(*rn*);
        varDepGraph[*rn*].add(*ln*);
      Append (*ln*, *c*) to *eqCons*;
  Append *c* to *otherCons*;
Initialize *coreVars* ← {*root_var*}, *Q* ← [*root_var*];
while *Q* ≠ ∅ **do**
  v ← dequeue *Q*;
  foreach *neighbor nb in varDepGraph*[*v*] **do**
    if *nb* ∉ *coreVars* **then**
      add *nb* to *coreVars*; enqueue *nb* onto *Q*;
Initialize *coreCons* ← [];
foreach (*ln*, *c*) *in eqCons* **do**
  if *ln* ∈ *coreVars* **then**
    Append *c* to *coreCons*;
foreach *constraint c in otherCons* **do**
  if *any v in getVars(c) has v.name* ∈ *coreVars* **then**
    Append *c* to *coreCons*;
Initialize *coreVarDomains* ← {};
foreach *name n in coreVars* **do**
  if *n* ∈ *varSorts* **then**
    coreVarDomains[*n*] ← *varSorts*[*n*];
return (*coreCons*, *coreVarDomains*);

Algorithm 3, the system builds a variable dependency graph, partitions equality and non-equality constraints, and performs backward traversal to extract a minimal constraint core.

In simple terms, Algorithm 3 proceeds as follows:

(1) First, we collect all constraints and extract the variables involved in each.
(2) For equality constraints (e.g., `CAR = OC / RC`), we treat them as bidirectional dependencies and record both sides in a graph.
(3) We then perform a backward traversal from the selected root variable (e.g., `capital_grade`), following all c.0
(4) Once the subgraph is collected, we extract only the constraints and variable domains that are within this cone.
(5) The resulting core set allows us to reason locally about how to flip a target variable or understand its dependency chain without solving the entire model globally.

This reduced core reveals the constraints and variable domains affecting the target, enabling localized inference and interpretable legal adjustments.

3.8 Soft Constraint Evaluation and Model Checking

We verify the satisfiability of soft and hard constraints using an SMT solver to ensure legal consistency. If the result is `sat`, the solver returns compliant values for soft constraints, which are converted to Python types via a `to_python` helper. If `unsat`, an *unsat core* reveals the minimal conflicting subset, guiding necessary revisions.

As summarized in Algorithm 4, this procedure safeguards against ill-formed logic and enables iterative refinement of scenario assumptions.

Algorithm 4: Soft Constraint Evaluation

Input: Optimize instance opt, list of soft constraints $(label, expr, expected)$
Output: Suggested assignments for soft constraints or unsat core

```
res ← opt.check();
if str(res) = "sat" then
    m ← opt.model();
    Function to_python(val) is
        if val = true then
            return True
        if val = false then
            return False
        s ← val.as_decimal(10);
        if s is an imprecise decimal representation then
            remove the uncertainty
            return float(s);
        return val;
    // Evaluate each soft constraint
    foreach (label, expr, expected) in softs do
        actual_z3 ←
          m.eval(expr, model_completion=True);
        actual ← to_python(actual_z3);
        print(label : default=expected, suggested=actual);
else if str(res) = 'unsat' then
    core ← opt.unsat_core();
    foreach c in core do
        print(c);
else
    print("Unknown result: " + res);
```

3.9 A Running Example on Insurance Regulation

To concretely demonstrate our system, we present a real-world case study that fully follows the end-to-end pipeline depicted in Figure 1. This example covers each module—from retrieval and SMT construction to compliance evaluation and interactive refinement—executed precisely as described in Section 4.

In our demonstration, we employ an official penalty case issued by Taiwan's Financial Supervisory Commission (FSC) against *Mercuries Life Insurance Co., Ltd.* as a real-world test scenario. This case was selected due to its richness in legal references, structured regulatory measures, and complex improvement planning elements, which align well with the full capabilities of our system pipeline. Through this example, we show how the system retrieves and parses legal constraints, constructs SMT logic for compliance modeling, evaluates statutory violations, and facilitates iterative plan refinements in a transparent and explainable manner. For full details and legal context of the case, please refer to **Appendix B**.

All components, including the LLM prompts, hybrid retrieval engine, SMT encoding logic, and GUI interface, are designed to allow legal professionals and researchers to test and extend the system in practice. The entire framework will be released as an open-source toolkit at https://github.com/shiayongshen/Automated-Legal-Analysis-for-Financial-Regulatory-Compliance.

3.9.1 Build SMT Constraints. To demonstrate our SMT encoding approach, we adopt the official sanction case issued by the Financial Supervisory Commission (FSC) against *Mercuries Life Insurance Co., Ltd.*, which is governed by the Insurance Act and the Capital Adequacy Management Act (CAMA). This case provides a concrete legal and factual context for encoding regulatory requirements and evaluating compliance. In our encoding, statutory obligations—such as minimum capital adequacy thresholds and improvement plan deadlines—are modeled as **hard constraints**, while case-specific facts—such as proposed remedial actions and projected ratios—are represented as **soft constraints**.

Insurance Act §143–4(1).

$$\text{CAR} = \frac{\text{OC}}{\text{RC}} \times 100 \quad \wedge \quad \text{CAR} \geq \theta_{\text{CAR}} \tag{6}$$

Capital Adequacy Rules §4–5.

$$\theta_{\text{CAR}} = 200 \tag{7}$$

$$\text{CapitalGrade} = \begin{cases} 1, & \text{CAR} \geq 200 \land \text{NWR} \geq 3 \\ 2, & 150 \leq \text{CAR} < 200 \\ 3, & 50 \leq \text{CAR} < 150 \\ 4, & \text{CAR} < 50 \lor \text{NWR} < 0 \end{cases} \tag{8}$$

Insurance Act §143–6.

$$\begin{aligned} &(\text{Grade} = 2) \land \neg \mathit{Plan} \rightarrow \mathit{AnyMeasure}_1 \\ &(\text{Grade} = 3) \land \neg \mathit{Plan} \rightarrow \mathit{AnyMeasure}_1 \land \mathit{AnyMeasure}_2 \\ &(\text{Grade} = 4) \land \neg \mathit{Plan} \rightarrow \mathit{AnyMeasure}_1 \land \mathit{AnyMeasure}_2 \land \mathit{Art149} \end{aligned} \tag{9}$$

Capital Adequacy Rules §2–3.

$$\text{RC} = \begin{cases} \text{Asset} + \text{Insurance} + \text{Interest} + \text{Other}, & \text{(Life)} \\ \text{Asset} + \text{Credit} + \text{Underw.} + \text{ALM} + \text{Other}, & \text{(Non-life)} \end{cases} \tag{10}$$

$$\text{OC} = \text{Tier1}_{\text{Unres}} + \text{Tier1}_{\text{Res}} + \text{Tier2} \tag{11}$$

Improvement Plan.

$$\begin{aligned} \text{Impr} &= \left(\frac{\text{OC} + \Delta\text{Cap}}{\text{RC}} \times 100 \right) - \text{CAR} \\ \Delta\text{Cap} &= \Delta\text{Cap}_{\text{Base}} + \Delta\text{Cap}_{\text{Private}} \end{aligned} \tag{12}$$

These constraints are evaluated using **Algorithm 4**, which automatically identifies legal violations and computes minimal corrections based on flipped soft constraint values.

3.9.2 Variable Refinement via Fixed Values and Bounds. When solver results diverge from domain expectations, users can refine variables through fixed values or bounded ranges. For example, if the automatically inferred risk capital of 100 units is considered unrealistic, a user may impose:

$$\text{RiskCapital} = 120$$

Alternatively, the following regulatory-informed bounds may be applied:

$$\begin{aligned} &30 \leq \text{Tier1}^{\text{unrestricted}} \leq 50 \\ &20 \leq \text{Tier1}^{\text{restricted}} \leq 70 \\ &40 \leq \text{Tier2} \leq 60 \end{aligned}$$

These values are then reprocessed as new hard constraints. The solver is guaranteed to return compliant configurations—if they exist—within these limits, offering practical adjustability in legal modeling.

3.9.3 Directed Inference Core on Capital Adjustment. To analyze the impact of a specific variable, we apply Algorithm 3 to perform dependency tracing via the Cone of Influence (COI) technique. In this case, we select the variable `pre_self_capital` as the root. Through the algorithm, we extract all downstream formulas and constraints that are directly or indirectly influenced by this variable.

The resulting subset of the SMT logic includes the following:

(1) $\text{pre\_CAR} = \frac{\text{pre\_self\_capital}}{\text{risk\_capital}} \times 100$
(2) $\text{final\_CAR} = \frac{\text{pre\_self\_capital+added\_capital}}{\text{risk\_capital}} \times 100$
(3) $\text{improvement\_plan} = \text{final\_CAR} - \text{pre\_CAR}$
(4) $\text{improvement\_plan} = \text{base\_improvement+private\_improvement}$
(5) $\text{pre\_capital\_grade} = \textbf{If}\ (\text{pre\_CAR} \geq 200 \land \text{net\_worth\_ratio} \geq 3,\ 1,\ \ldots)$
(6) $\text{capital\_grade} = \textbf{If}\ (\text{final\_CAR} \geq 200 \land \text{net\_worth\_ratio} \geq 3,\ 1,\ \ldots)$
(7) $\text{final\_CAR} \geq 200$

From this extracted logical context, we can clearly observe how `pre_self_capital` influences a cascade of downstream variables and thresholds. This type of localized reasoning not only enhances interpretability, but also provides a precise understanding of the variable's role within the SMT-based optimization environment. The greater the number and depth of dependent constraints, the more central the variable is to the decision process.

4 Experiment

To validate our framework, we conducted an empirical analysis based on 87 real-world penalty cases issued by the Financial Supervisory Commission (FSC) of Taiwan. In addition, we developed an interactive user interface using Gradio to facilitate real-time interaction with the system. As part of our infrastructure, we also curated a comprehensive regulatory database, comprising 75 distinct laws and 3,753 statutory provisions extracted from the Taiwan Law Database.

This experiment is designed to address the following research questions:

- **RQ1**: Can LLM generate SMT well?
- **RQ2**: Can LLM+SMT outperform LLM in reasoning illegal terms in real cases?
- **RQ3**: Can LLM+SMT outperform LLM in reasoning correction in real cases?

We evaluate these questions through both qualitative and quantitative assessments, focusing on reasoning performance, legal alignment, and correction accuracy.

4.1 SMT Constraint Synthesis

We conducted generation experiments on 87 regulatory cases. Using a few-shot learning approach, we prompted the LLM to generate Z3 SMT code tailored to each case. The generated code was then manually reviewed to verify its logical soundness and contextual correctness. If the code was valid, it was categorized as *correct*; otherwise, we prompted the model to iteratively revise its output until the issues were resolved, and such cases were classified as *incorrect*.

Common errors included omissions of impacted clients in compliance outcomes, failure to zero out fines where applicable, and logically inconsistent constraint sets resulting in `unsat` responses from the solver. Through manual correction and validation, we ensured that all final SMT codes were accurate and could serve as ground truth for downstream evaluation.

The generation accuracy and response times are summarized in Table 1.

Table 1: Summary of LLM-generated SMT code

Method	Accuracy	Unsat Rate	Avg. Time (s)
LLM Generate SMT Code	0.8621	0.0230	36.85

4.2 Illegal Term Reasoning

Since our system is designed to analyze real-world legal cases, one of our core objectives is to identify the specific illegal terms embedded within the scenario. By highlighting these illegal terms, we aim to help users better understand which actions must be taken to comply with relevant regulations and avoid penalties. To evaluate this capability, we compared the number of illegal terms identified and the average response time between the baseline LLM and our proposed LLM+SMT approach, as shown in Table 2.

Table 2: Illegal term analysis: Comparison of LLM and LLM+SMT.

Method	Avg. Illegal Terms per Case	Avg. Time (s)
LLM + SMT	5.08	0.021
LLM Only	6.40	7.01

To explain how these illegal terms are identified, our SMT-based component leverages the `assert` and `track` commands during the code generation phase. When a conflict is detected (i.e., an `unsat` result), it indicates a contradiction between the statutory requirements and the actual behavior described in the case. For example, if the law mandates certain remedial actions in response to capital inadequacy, but the case fails to mention such actions, this mismatch leads to an `unsat` state.

By default, Z3 returns only a minimal unsatisfiable core, which may omit other relevant violations. To address this limitation, we treat all legal constraints—including originally defined soft constraints—as hard constraints within the solver. When conflicts arise, the solver returns multiple unsat cores. We then collect the components of each unsat core that originated from hard constraints and define them as the "illegal terms." The remaining parts, originally from soft constraints, are converted into free variables and reintroduced iteratively until the system reaches a satisfiable state. The number of unique illegal terms is computed using set operations across these iterations.

To further validate our approach, we also queried the LLM directly with the same case and legal text to identify illegal terms. However, the reasoning output from the LLM alone lacks interpretability and cannot guarantee logical consistency. In contrast, our SMT-enhanced method not only achieves comparable illegal term identification but also significantly outperforms the LLM in terms of response time, offering a 333× speed improvement while providing greater explainability and reliability.

For instance, in a real-world case involving an insurance company in appendix-A, our SMT-based system identified the following set of violated constraints through five rounds of unsatisfiability analysis, completed in just 0.191 seconds:

- `req_net_worth`
- `self_capital_sum`
- `risk_capital_sum`
- `def_improvement_plan`
- `req_final_CAR`
- `def_pre_CAR`
- `def_total_improvement`

These constraints represent formal statutory violations that must be resolved for the case to become compliant. In contrast, the LLM required approximately 5 seconds to generate a longer list of over ten issues, including vague or overlapping statements such as:

- Capital Adequacy Ratio below statutory level
- Net Worth Ratio below regulatory requirement
- Significant capital inadequacy
- Failure to submit a complete and feasible capital improvement plan
- Failure to ensure capital adequacy level is met in 2024
- High uncertainty in planned private placement of hybrid instruments
- Inadequate backup plan
- Violation of Insurance Act Article 143-6, Paragraph 1, Subparagraph 1
- Restricted transactions with related parties
- Noncompliance with capital adequacy regulations

Closer inspection reveals that the additional terms generated by the LLM-only approach tend to fall into the following three categories:

(1) **Contextual Misclassifications**: These are contextually relevant but legally non-binding observations. For example, “high uncertainty in planned private placement of hybrid instruments” reflects financial risk, but does not constitute a statutory violation.
(2) **Misinterpretation of Advisory Clauses**: The LLM sometimes elevates soft recommendations or best practices to mandatory obligations. For instance, the lack of a contingency plan may be flagged as illegal even if it is not legally required.
(3) **Vague or Composite Labels**: Statements such as “noncompliance with capital adequacy regulations” are overly broad and fail to pinpoint a specific violated provision, reducing traceability and diagnostic precision.

These discrepancies underscore a key strength of our SMT-based framework: by explicitly modeling enforceable legal constraints and verifying them through symbolic reasoning, our method avoids ambiguity, ensures logical consistency, and isolates truly actionable statutory violations. In domains like financial regulation where interpretability, auditability, and precision are paramount, this difference offers a compelling practical advantage.

4.3 Correction Reasoning

Following the analysis of illegal term identification, we proceed to evaluate the models’ capabilities in correcting real-world cases. This task investigates what changes are necessary to revise a case such that the resulting legal outcome would be penalty-free. We performed this correction analysis on all 87 cases.

For the SMT-based approach, we increased the weight of the penalty-related soft constraints and set the target penalty value to zero. This allows the SMT solver to return a satisfiable assignment that eliminates the penalty, which we define as the *ground truth* for comparison. We then compare this result with the correction output produced by the LLM.

In this experiment, the LLM is provided with the soft constraint variables and their original values. It is allowed to freely decide

whether to flip the value of a given variable or assign a new value. The LLM is also given the corresponding case text and statutory references to support its reasoning process.

To further evaluate the robustness of the models, we randomly selected 24 cases and introduced contradictory constraints into the solver. For example, we may define a capital adequacy ratio requirement of over 200%, where the capital adequacy is calculated as own capital divided by risk-weighted capital. If we then add a constraint that sets own capital to zero, the system becomes unsatisfiable. These newly introduced constraints are also provided to the LLM. If the LLM is able to reason that the problem has no feasible solution, it should return an unsatisfiable judgment.

The performance comparison is shown in Table 3.The confusion matrix for the LLM-only baseline is shown in Table 4.

Table 3: Performance of case correction reasoning.

Method	Accuracy	Precision	Recall	F1 Score	Avg. Time (s)
LLM + SMT	1.0000	1.0000	1.0000	1.0000	0.0040
LLM Only	0.3333	0.6190	0.2063	0.3080	1.4700

Table 4: Confusion matrix for LLM without SMT Solver.

Prediction	Actually Satisfiable	Actually Unsatisfiable
Satisfiable	13 (TP)	8 (FP)
Unsatisfiable	50 (FN)	16 (TN)

From the results, we observe that the LLM tends to default to predicting an unsatisfiable outcome, even for cases where a valid solution exists. This bias leads to a significant number of false negatives. Furthermore, the LLM's correction reasoning lacks consistency and completeness compared to the SMT solver. In contrast, the SMT-based method provides not only perfect accuracy but also significantly faster inference, completing each correction in approximately 0.004 seconds—over 300 times faster than the average LLM response time.

Summary of Experimental Findings

Across all three research questions, the experimental results consistently demonstrate the effectiveness of incorporating SMT solvers into the legal reasoning pipeline. For **RQ1**, the LLM successfully generated syntactically and semantically correct SMT code in over 86% of cases, enabling precise modeling of legal constraints. In **RQ2**, the SMT-enhanced method not only produced comparable illegal term identification to the baseline LLM but also offered significantly faster response times (333x speedup), with greater logical clarity and explainability. Finally, in **RQ3**, the correction reasoning powered by LLM+SMT achieved perfect accuracy, outperforming the baseline LLM across all major classification metrics, including precision, recall, and F1 score. Moreover, the SMT solver demonstrated superior robustness in handling contradictory constraints and provided interpretable correction paths that reflect regulatory compliance. Collectively, these findings affirm that the hybrid LLM+SMT framework is not only feasible but also advantageous for real-world legal reasoning and corrective decision-making.

5 Conclusion

This study proposes a hybrid compliance analysis framework that integrates Large Language Models (LLMs) with Satisfiability Modulo Theories (SMT) solvers to address the increasing complexity and frequent revisions of financial regulations in Taiwan. By encoding legal statutes as hard constraints and case-specific facts as soft constraints, the system not only detects non-compliance but also provides minimal corrective suggestions to restore legality.

Furthermore, we ensure that the processes of Cone-of-Influence (COI) analysis and boundary setting can be repeated iteratively until users fully understand the reasoning behind a given case. This design promotes transparency and interpretability in legal diagnostics. To enhance accessibility and user interaction, the entire framework is deployed on a Gradio interface, allowing users to explore compliance evaluations and constraint-based reasoning through an intuitive and interactive web application.

The results of empirical evaluations on 87 real-world penalty cases issued by Taiwan's Financial Supervisory Commission (FSC) demonstrate that integrating SMT solvers significantly improves the legal reliability and reasoning accuracy of LLM-generated outputs. This hybrid approach lays a strong foundation for extending formal verification tools to broader regulatory domains and advancing trustworthy, AI-driven legal and financial analysis.

This study proposes a hybrid compliance analysis framework that integrates Large Language Models (LLMs) with Satisfiability Modulo Theories (SMT) solvers to address the increasing complexity and frequent revisions of financial regulations in Taiwan. By encoding legal statutes as hard constraints and case-specific facts as soft constraints, the system not only detects non-compliance but also provides minimal corrective suggestions to restore legality.

Furthermore, we ensure that the processes of Cone-of-Influence (COI) analysis and boundary setting can be repeated iteratively until users fully understand the reasoning behind a given case. This design promotes transparency and interpretability in legal diagnostics. To enhance accessibility and user interaction, the entire framework is deployed on a Gradio interface, allowing users to explore compliance evaluations and constraint-based reasoning through an intuitive and interactive web application.

The results of empirical evaluations on 87 real-world penalty cases issued by Taiwan's Financial Supervisory Commission (FSC) demonstrate that integrating SMT solvers significantly improves the legal reliability and reasoning accuracy of LLM-generated outputs. This hybrid approach lays a strong foundation for extending formal verification tools to broader regulatory domains and advancing trustworthy, AI-driven legal and financial analysis.

While the current implementation focuses on Taiwanese insurance law, the proposed architecture is designed with cross-jurisdictional adaptability in mind. The clause parser and legal encoder are modular, enabling straightforward adaptation to other legal systems by updating parsing templates and statutory mappings. Additionally, the retrieval-augmented generation (RAG) component and SMT solver are fully decoupled, requiring only minimal changes—such as replacing the corpus and prompt patterns—to support multilingual and international use cases. Given the extensive legal text coverage in LLM pretraining corpora for English and

European languages, we anticipate even greater generalization performance in Western regulatory contexts. These features highlight the system's strong potential for international adoption and legal compliance automation across diverse jurisdictions.

Table 5: Internationalization Potential: Minimal Engineering Modifications Required for Legal Domain Transfer

Module	Purpose	Requires Modification?
RAG Corpus	Retrieve relevant legal text and precedents	**Yes** (replace with local corpus)
Prompt Template	Control query format and language context	**Yes** (adjust language and legal phrasing)
Clause Parser	Map legal clauses to logical templates	**Partial** (redefine parsing patterns)
Legal Mapping Dictionary	Link clause labels to statutes	**Yes** (update to target jurisdiction)
Z3 Rule Engine	Enforce symbolic legal constraints	**No** (fully reusable)
SMT Relaxation Loop	Perform minimal correction search	**No** (domain-agnostic logic)
Gradio UI	Provide user interface	**No** (language-localizable)

References

[1] Holger Andreas, Matthias Armgardt, and Mario Gunther. "Counterfactuals for causal responsibility in legal contexts". In: *Artificial intelligence and law* 31.1 (2023), pp. 115–132.

[2] Douglas W Arner, Janos Barberis, and Ross P Buckley. "The evolution of Fintech: A new post-crisis paradigm". In: *Geo. J. Int'l L.* 47 (2015), p. 1271.

[3] S Athul et al. "LegalMind System and the LLM-based Legal Judgment Query System". In: *2024 International Conference on Trends in Quantum Computing and Emerging Business Technologies*. IEEE. 2024, pp. 1–5.

[4] Clark Barrett et al. "cvc4". In: *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings 23*. Springer. 2011, pp. 171–177.

[5] Thomas H Beach et al. "A rule-based semantic approach for automated regulatory compliance in the construction sector". In: *Expert Systems with Applications* 42.12 (2015), pp. 5219–5231.

[6] Armin Berger et al. "Towards Automated Regulatory Compliance Verification in Financial Auditing with Large Language Models". In: *2023 IEEE International Conference on Big Data (BigData)*. 2023, pp. 4626–4635. DOI: 10.1109/BigData59044.2023.10386518.

[7] Richard J Bolton and David J Hand. "Statistical fraud detection: A review". In: *Statistical science* 17.3 (2002), pp. 235–255.

[8] Curtis A Bradley and Trevor W Morrison. "Presidential Power, Historical Practice, and Legal Constraint". In: *Columbia Law Review* (2013), pp. 1097–1161.

[9] Leonardo De Moura and Nikolaj Bjørner. "Z3: An efficient SMT solver". In: *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2008, pp. 337–340.

[10] Niklas Eén and Niklas Sörensson. "An extensible SAT-solver". In: *International conference on theory and applications of satisfiability testing*. Springer. 2003, pp. 502–518.

[11] Tom Fawcett and Foster Provost. "Adaptive fraud detection". In: *Data mining and knowledge discovery* 1.3 (1997), pp. 291–316.

[12] Nick Feng et al. "Early verification of legal compliance via bounded satisfiability checking". In: *International Conference on Computer Aided Verification*. Springer. 2023, pp. 374–396.

[13] Samuel Judson et al. "soid: A Tool for Legal Accountability for Automated Decision Making". In: *International Conference on Computer Aided Verification*. Springer. 2024, pp. 233–246.

[14] Patrick Lewis et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474.

[15] Tomer Libal and Tereza Novotná. "Towards transparent legal formalization". In: *Explainable and Transparent AI and Multi-Agent Systems: Third International Workshop, EXTRAAMAS 2021, Virtual Event, May 3–7, 2021, Revised Selected Papers 3*. Springer. 2021, pp. 296–313.

[16] Sara Ebrahim Mohsen, Allam Hamdan, and Haneen Mohammad Shoaib. "Digital transformation and integration of artificial intelligence in financial institutions". In: *Journal of Financial Reporting and Accounting* (2024).

[17] Eric WT Ngai et al. "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature". In: *Decision support systems* 50.3 (2011), pp. 559–569.

[18] Igor Ponkin and Alena Redkina. "Digital formalization of law". In: *International Journal of Open Information Technologies* 7.1 (2019), pp. 39–48.

[19] Mohammad Raza and Natasa Milic-Frayling. "Almost Sure Reasoning: Generating Verified Formalizations with Language Models and Logical Solvers". In: ().

[20] Dong Shu et al. "LawLLM: Law Large Language Model for the US Legal System". In: *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2024, pp. 4882–4889.

[21] Harry Surden. "Computable Law and Artificial Intelligence". In: *U of Colorado Law Legal Studies Research Paper Forthcoming, Cambridge Handbook of Private Law and Artificial Intelligence (forthcoming 2024)* (2024).

[22] Christos Tsigkanos et al. "How do firms transact? Guesstimation and validation of financial transaction networks with satisfiability". In: *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE. 2019, pp. 15–22.

[23] Sandra Wachter, Brent Mittelstadt, and Chris Russell. "Counterfactual explanations without opening the black box: Automated decisions and the GDPR". In: *Harv. JL & Tech.* 31 (2017), p. 841.

A Taiwan's Insurance Law

A.1 Taiwan's Insurance Act No.143-6

(Competent Authority's Adopting Relevant Supervisory Measures, Subject to the Insurance Enterprise's Capital Adequacy Ratio) The competent authority shall adopt any or all of the following measures, subject to the grading of an insurance enterprise's capital:

1. In the case of inadequate capital:

(1) To order the insurance enterprise and its responsible person to put forward a plan for capital increase or another corrective action plan for finance or business within the specified period; where the insurance enterprise fails to submit the same within the specific period or fails to implement the plan, the competent authority may take supervision measures against the capital adequacy ratio;

(2) To order the insurance enterprise to cease selling insurance products or restrict it from launching new insurance products;

(3) To restrict the scope of fund utilization;

(4) To restrict the remuneration, bonus, stock options, or other payments of similar properties to the responsible person of the insurance enterprise;

(5) To take other necessary measures.

2. In the case of significantly inadequate capital:

(1) To take the measures set forth in the preceding subparagraph;

(2) To dismiss the responsible person of the insurance enterprise, and inform the authority in charge of company registration to revoke the registration of the insurance enterprise;

(3) To suspend the responsible person from exercising his/her duties within a certain period;

(4) To require the insurance enterprise to obtain prior approval by the competent authority itself for acquisition or disposal of any specific assets;

(5) To order the insurance enterprise to dispose of the assets specified;

(6) To restrict or forbid the insurance enterprise to extend loans to or conduct other transactions with the related parties thereof;

(7) To order the insurance enterprise to lower the remuneration of its responsible person, which, afterward, shall not exceed 70% of the average remuneration paid to said responsible person within the 12 months before the insurance enterprise's capital is graded as being significantly inadequate;

(8) To restrict establishment of or order dissolution, within the specified period, of a branch or department; and

(9) To take other necessary measures.

3. In the case of seriously inadequate capital: In addition to the measures referred to in the preceding subparagraph, the measures

referred to in the sub-paragraph 1 of paragraph 3 of Article 149 herein shall also apply.

B Financial Penalty Issued by Taiwan's Financial Supervisory Commission (FSC)

- **Summary:** The FSC imposed a regulatory sanction on Mercuries Life Insurance Co., Ltd. for failing to meet the statutory capital adequacy ratio and net worth ratio as stipulated by the Insurance Act. Despite submitting a capital, financial, and business improvement plan, the company was unable to elevate its capital adequacy ratio to the legally required threshold for fiscal year 2024. Pursuant to Article 143-6, Paragraph 2, Subparagraphs 1 and 6 of the Insurance Act, the FSC ordered the company to: (1) suspend all new credit or transactions with related parties (excluding existing or renewed contracts), until the company meets the capital adequacy standard and obtains FSC approval; and (2) submit a concrete, complete, and legally compliant improvement plan within one month.
- **Details:**
  (1) Date of Sanction: July 11, 2024
  (2) Penalized Entity: Mercuries Life Insurance Co., Ltd.
  (3) Company Representative: Mr. Weng (full name redacted)
  (4)-(7) Personal identity information such as date of birth, gender, ID number, and address are omitted.
  (8) Summary of Violation: As of the end of 2023, the company's capital adequacy ratio (111.09%) and net worth ratio (2.97%) were below the minimum standards set by the "Regulations Governing Capital Adequacy of Insurance Enterprises." The company submitted a plan with six improvement measures, including reclassification of real estate assets, cash capital increase of 300 million shares, issuance of subordinated debt, and private placement of preferred stocks. However, the FSC deemed these plans deficient due to lack of contractual commitments, uncertain timelines, and insufficient projected improvements in capital adequacy. Even if fully implemented, the projected increase in capital adequacy would not meet the minimum standard of 200%.
  (9) Findings and Rationale:
  – **113-year (2024) Plan:** The projected capital adequacy improvement (about +17.9% to +30.4%) was deemed insufficient to meet the statutory threshold.
  – **114-year (2025) Plan:** The measures were considered too delayed and uncertain to address current deficiencies.
  – The proposed contingency plan would only activate in 2025, failing to remedy the 2024 capital shortfall.
  (10) Legal Basis:
  – According to Article 143-4, Paragraphs 1 and 4 of the Insurance Act and Article 4 of the "Regulations Governing Capital Adequacy of Insurance Enterprises," insurers must maintain minimum capital adequacy ratios. The company's current ratios fall under the classification of "Significantly Inadequate Capital."
  – The improvement plan lacked feasibility, specific timelines, and enforceable mechanisms, and thus was deemed incomplete and in violation of Article 143-6, Paragraph 1, Subparagraph 1.
  – In consideration of the company's past efforts (e.g., asset disposal, equity sales, four capital increases totaling NT$13.751 billion), and the reported net worth ratio improvement to 3%, the FSC decided to impose a conditional restriction as per Article 143-6, Paragraph 2, Subparagraph 6.
  The company must submit a revised and concrete capital improvement plan within one month from the effective date of this decision. Non-compliance may result in further regulatory action.
  Note: If the company objects to this sanction, it may file an administrative appeal within 30 days from the day following receipt of this decision, pursuant to Article 58 of the Petition and Appeal Act. However, under Article 93 of the same Act, the filing of an appeal does not suspend enforcement of the sanction unless otherwise specified by law.
- **Date:** July 12, 2024
- **Source:** `https://www.fsc.gov.tw/ch/home.jsp?id=131&parentpath=0,2&mcustomize=multimessages_view.jsp&dataserno=202407120004&dtable=Penalty`

C LLM Prompt Template

C.1 Extra Laws Article

You are a legal expert. Based on the input case, please identify the relevant legal statutes and article numbers. Please ignore the Petition Act.

```
{
 "LegalCase": "A description of the legal incident goes here.",

  "RelevantRegulations": [
    {
      "RegulationName": "Company Act",
      "Articles": [15, 20]
    },
    {
      "RegulationName": "Securities and Exchange Act",
      "Articles": [3, 7]
    }
  ]
}
```

C.2 Query rewrite

Please analyze the following legal provision and identify any potentially unclear or underspecified parts.

Generate 3 to 5 search questions targeting these ambiguities in order to locate supplementary explanations or formal definitions. **Do not generate more than 5 questions.**

Legal Text: `{law_text}`

Please pay special attention to the following aspects:

(1) Definitions or calculation methods for technical terms mentioned in the provision
(2) Specific threshold values or regulatory standards referenced

(3) Clarification of which institutions are being referred to (e.g., "insurance enterprises" in insurance laws, or "banks" in banking laws)
(4) Only output the list of questions. Each question should be concise, precise, and phrased as a complete sentence.

C.3 Legal Compliance Explanation

The following is the output of a Z3 Solver execution for a regulatory violation case. If the result is `sat`, it indicates that by modifying the soft constraints, the scenario can be legally remediated. The flipped variables represent the elements that the company can potentially improve.

Solver Execution Result: `{execution_result_str}`
Relevant Legal Provisions: `{law_article}`

Please analyze the flipped variables and explain, based on the legal provisions, what concrete actions the company can take to bring each flipped variable into compliance. Focus on:

- Legal reasoning that justifies why the variable must be changed
- Specific operational or financial measures that can be taken
- How these actions align with the obligations described in the relevant legal provisions

Your output should be a list of flipped variables, each followed by a short paragraph explaining possible remedies or adjustments that comply with the law.

C.4 Legal Compliance Explanation with Fixed Value and Bounds

The following information pertains to a specific financial regulatory violation case analyzed using Z3 SMT Solver. The user has chosen to convert selected soft constraints into hard constraints.

Original Case Description:

`{case_content_str}`

Original Z3 Execution Output:

`{original_execution_result}`

User-Imposed Hard Constraints (with specified bounds or values): modified_constraints

Solved Values of the Hard-Constrained Variables (if satisfiable): solved_values_str

Z3 Execution Output After Hard Constraint Injection:

`{execution_result}`

Relevant Legal Provisions:

`{law_article}`

Please analyze the updated scenario based on the provided information. Your output should address the following:

(1) Explain the legal and model-level impact of converting each listed variable from a soft constraint to a hard constraint.
(2) For each affected variable, provide detailed legal interpretation and practical explanation of how increasing or decreasing the value may improve compliance.
(3) Analyze the correlation between legal statutes and numeric results. Identify potential risks or infeasibilities from both legal and financial perspectives.
(4) Highlight flipped variables, and explain why each change occurred and which legal clause(s) it is linked to.
(5) Offer actionable compliance recommendations, especially those involving adjustments to capital structure or risk metrics. Explain what insights are derived from the new variable values.
(6) For each hard-constrained variable, clearly describe whether its solved value satisfies the user's goal. If not, explain the gap and suggest a viable path to achieve compliance.

Your response should emphasize detailed numerical analysis and value comparison. Discuss how solver results align or diverge from user-defined targets. This numeric-to-legal mapping is a key feature of the tool.

C.5 Illegal Term Identification

You are now a legal expert. Based on the given case and relevant legal provisions, generate a list of illegal terms. Please present the list as a Python `list` with Traditional Chinese-English mixed descriptions.

Please note the following instructions:

- Your conclusions must be strictly based on legal reasoning derived from the provided provisions; do not speculate arbitrarily.
- If uncertain, do not generate the term.
- Carefully consider before responding.

Case: `{case}`
Legal provisions: `{softs}`

C.6 Case Correction Reasoning

You are a constraint rule design assistant.

Given a list of soft constraint variables considered by the model during optimization, select a subset of these variables that you consider important or reasonable, and assign specific values to convert them into hard constraints.

These hard constraints will eliminate the associated penalties.
Example case: `{case}`
Relevant legal provisions: `{law}`
Soft constraints: `{softs}`
Additional variables: `{add_constraints}`
Please adhere to the following rules:

- Do not set all variables directly to their expected values.
- Select only a subset of variables to convert into hard constraints and assign recommended values.
- Do not directly set penalty variables (e.g., `fine_total`, `fine_amount`) to zero.
- Return only a JSON-formatted result without additional explanation.
- Do not add new variables; use only the provided variables.
- If under the additional conditions it is impossible to eliminate the penalties, respond with `"no solution"` instead of JSON.

Return format example:

```
{
  "kyc_violation": true,
  "amount_affected": 20000000,
  "cust_adj": 2
}
```

If no solution is possible, return `"no solution"`.

D LegalSearchEngine Module

Embedding Model `text-embedding-ada-002` — OpenAI model for generating semantic embeddings.

Vector Database `ChromaDB` — Persistent vector database for storing legal articles.

Keyword Search `BM25Okapi` — Lexical retrieval method to complement semantic results.

Reranker Model `BAAI/bge-reranker-v2-m3` — Used to rank results based on semantic relevance.

Tokenization Character-level tokenizer tailored for Chinese text.

Hybrid Scoring Weighted combination: 0.8· vector score + 0.2· BM25 score.

API Key Config-secured OpenAI API key via `OPENAI_API_KEY`.

Interface Supports both command-line and interactive querying.

Result Categorization Classifies articles as direct or indirect matches (threshold: 0.75).

Logging Python `logging` for monitoring runtime behavior.

E System User Interface

Figure 4: Solver Result from Z3 and LLM-Generated Interpretations

Figure 3 shows the main interface for querying financial supervision cases. It consists of the following sections:

- **Top input area (left)**: The user enters a natural language query related to financial regulation (e.g., "Capital rate").
- **Result count slider (right)**: Allows users to select how many relevant cases they wish to retrieve from the knowledge base.
- **Case display area (bottom-left)**: Displays one or more retrieved legal cases that match the user's input. Each case includes real financial supervisory statements and relevant articles.
- **Code mapping area (bottom-right)**: Shows the corresponding Z3 constraint model used for reasoning on that case, including the financial formulas and assumptions tied to the retrieved scenario.

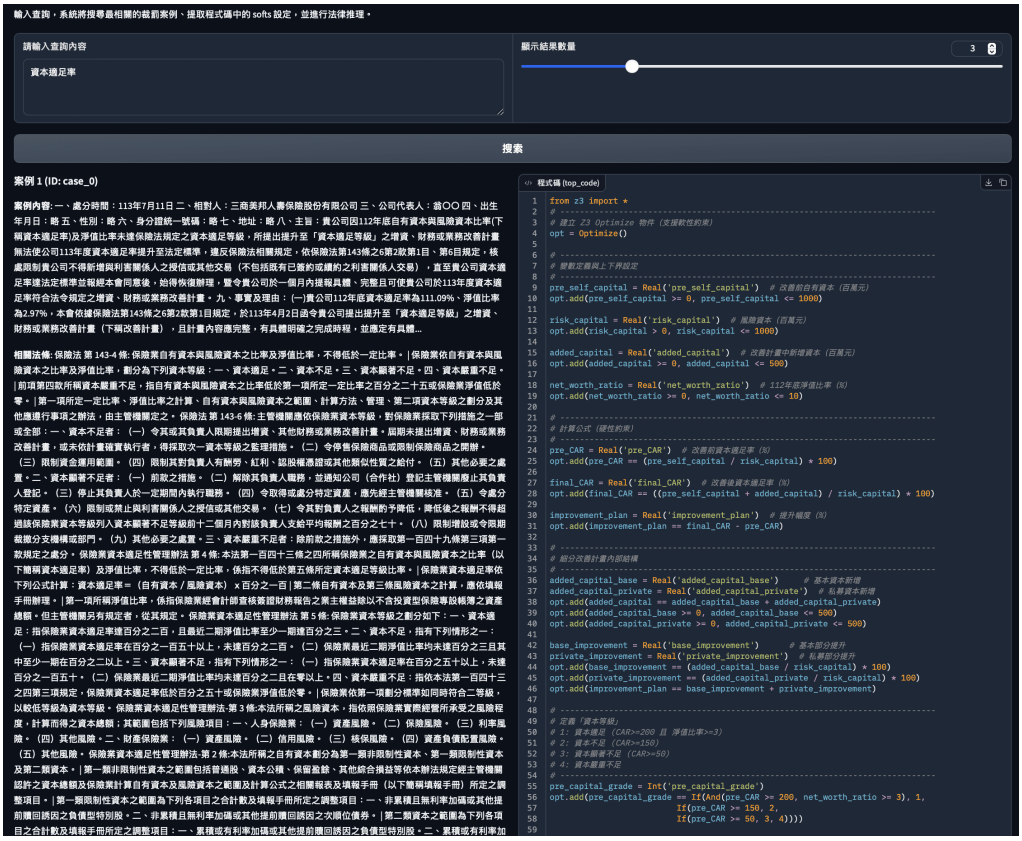

Figure 3: Case Query and Code Mapping Interface

Figure 4 shows the output from the system's backend Z3 solver, along with corresponding explanations generated by a large language model (LLM). This dual-layer interface provides both formal optimization results and human-readable recommendations:

- **Z3 Solver Output (Top)**: Displays the optimized values calculated by solving soft constraints using Z3. These include specific financial indicators such as `final CAR`, `pre_capital_grade`, or `improvement plan`, with both expected and suggested values based on constraint satisfaction.
- **LLM Interpretation (Bottom)**: The system automatically converts solver outputs into natural language recommendations. The LLM interprets each variable's role and regulatory meaning, then suggests practical policy actions that companies could take to improve their financial position or meet legal adequacy standards.

Figure 5 demonstrates how users can configure soft constraints in the system, either by:

- **Fixing a value** using the `hard` flag, which locks a variable to a specific value (as shown in the row with `use_bounds = false`).
- **Defining a value range (bound)** using the `lower_bound` and `upper_bound` fields, allowing the solver to optimize freely within user-approved thresholds.

In this example, three variables related to **private capital** were selected:

- First restricted capital
- First restricted investment
- First restricted reserve ratio

Users can flexibly combine fixed and bounded settings to guide the optimization process while preserving business constraints.

Soft Constraint

label	actual	expected	hard	use_bounds	lower_bound	upper_bound
人身保險業-其他風險（保資第3條第1款(四)）	0	0	false	true	0	0
人身保險業-利率風險（保資第3條第1款(三)）	0	0	false	true	0	0
人身保險業-資產風險（保資第3條第1款(一)）	0	0	false	true	0	0
具體指前時程及程序（False=無, True=有）	false	true	false	false		
基本提升比率	17.9	17.9	false	true	14.32	21.48
已履訂私募負債型特別股合約(False=未履, True=已履)	false	true	false	false		
年度計畫提升幅度	30.4	50	false	true	24.32	36.48
提交完整且可行的增資/財務/業務改善計畫（False=無, True=有）	false	true	false	false		
最終CAR	150	200	false	true	120	180
私募改善提升比率	12.5	32.1	false	true	10	15
第一類限制性資本（保資第2條）	0	0	true	false	0	0
第一類非限制性資本（保資第2條）	0	0	true	true	38	88
第二類資本（保資第2條）	0	150	true	true	40	90
自有資本	150	150	false	true	120	180
財產保險業-信用風險（保資第3條第2款(二)）	0	0	false	true	0	0
財產保險業-其他風險（保資第3條第2款(五)）	0	0	false	true	0	0
財產保險業-核保風險（保資第3條第2款(三)）	0	0	false	true	0	0
財產保險業-資產負債配置風險（保資第3條第2款(四)）	0	100	false	true	0	0
財產保險業-資產風險（保資第3條第2款(一)）	0	0	false	true	0	0
資本等級	2	1	false	false		
風險資本	100	100	false	true	88	120

Figure 5: User-defined Soft Constraints with Fixed Values and Bounds

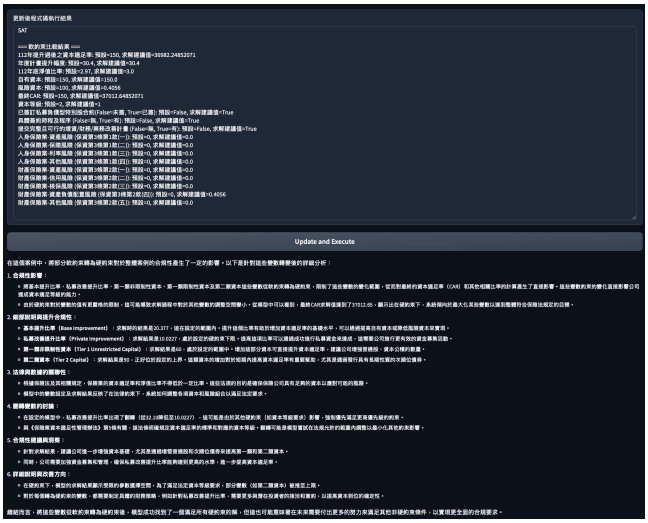

Figure 6: Solver Output under User-Defined Fixed or Bounded Constraints, with LLM Explanation

Figure 6 illustrates the output of the SMT solver (Z3) when specific soft constraints are either fixed (using the `hard` flag) or bounded (using `lower_bound` and `upper_bound`). The system responds by:

- **Solver Result (Top Panel)**: Based on the fixed/bounded constraints, Z3 computes the optimal values of related financial indicators.
- **LLM Interpretation (Bottom Panel)**: The system then uses a large language model (LLM) to generate natural-language recommendations. These explain how to improve financial adequacy based on the solver's output, including strategic suggestions across dimensions such as capital replenishment, legal risk mitigation, or internal operations.

Figure 7: Variable Dependency Analysis (DIC) Interface

Figure 7 illustrates the system's Directed Inference Core (DIC) analysis module. This interface enables users to trace the logical dependencies of a selected variable within the SMT formulation.

- **Variable Selection (Top)**: The user selects a variable (e.g., `Capital Grade`) to analyze from the dropdown menu.
- **DIC Result Panel (Bottom)**: The system computes and displays all relevant constraints, arithmetic expressions, and logical rules associated with the selected variable. These include:
  - Constraint equations (e.g., how `final_CAR`, `improvement_plan`, and `capital_grade` are calculated)
  - Logical implications and conditions based on variable values
  - Direct and transitive dependencies (e.g., If `capital_grade = 1` then policy A is enforced)

This functionality enhances transparency, debuggability, and explainability of the system's constraint logic.

Furthermore, it allows users to gain a deeper understanding of how a specific variable behaves within the overall constraint network, and to identify the necessary conditions under which the variable of interest can reach a legally compliant state.