Bridging LLMs and Symbolic Systems: A Deterministic Rule-based Layer for Reliable High, Stakes AI

Vincent A. Powell
vincent.powell@oblongix.com

Abstract

Large Language Models (LLMs) offer strong capabilities for automating complex decisions, but they exhibit inherent instability: identical prompts can yield different outputs, and generated content may occasionally be unsupported or incorrect. Such behaviour makes LLMs unsuitable as standalone decision engines in high-stakes domains such as banking, healthcare, or insurance where outcomes must be reproducible, explainable, and auditable.

We introduce SESL (Simple Expert System Language), a deterministic rule-based system that guarantees identical outputs for identical inputs. SESL expresses policies as human, readable rules, provides faithful explanations grounded in execution traces, and includes built, in tools for testing and validation.

We propose a hybrid architecture in which LLMs assist with rule authoring while SESL performs the actual decision execution. In an evaluation using 100 synthetic loan applications (each evaluated six times), SESL produced perfectly consistent results across all 600 evaluations, whereas the LLM, only approach showed both decision inconsistencies (3, 5% volatility) and systematic policy errors (42% error rate before prompt tuning, improving to 22% after tuning).

These findings suggest that combining LLM, assisted authoring with deterministic symbolic execution offers a practical pathway toward trustworthy AI in regulated settings.

1. Introduction

LLMs such as GPT, style models have demonstrated strong performance on knowledge, intensive tasks, natural language reasoning, and code generation. However, they are stochastic generative models: the same prompt can yield different outputs, and these outputs may include confident yet fabricated facts or justifications.

Recent empirical studies have documented hallucination rates of approximately 1.75% in user reported issues with AI powered mobile applications and other studies show that chain-of-thought explanations (the step, by, step reasoning that LLMs generate when solving problems) are unstable across samples and need not reflect the model's true reasoning process [4–6,15]. In other words, when an LLM shows its "work," that explanation may not accurately represent how it actually arrived at its answer.

As LLM deployments expand from isolated tasks to multi, stage or agentic workflows, the associated risks compound: errors introduced in one step can propagate, amplify, or interact with later reasoning stages, leading to cascading failures that are difficult to detect or audit [31].

Regulatory and governance frameworks, including the European Parliament AI Act [32], OECD AI Principles [7], and the NIST AI Risk Management Framework [8], emphasize transparency, accountability, and risk mitigation. The European Union's General Data Protection Regulation (GDPR) establishes requirements for meaningful information about algorithmic decision, making, while ongoing debates about a "right to explanation" [9,10,18] reflect increasing demands for interpretable AI systems. Organizations therefore require infrastructure that leverages the flexibility of LLMs while enforcing determinism and explainability.

These properties of LLMs conflict with the requirements of high, stakes environments such as credit risk modelling, fraud detection, clinical triage, and regulatory compliance, where decisions must be reproducible, auditable, and accompanied by clear justifications [16,17].

Recent efforts to improve the governance of LLMs such as guardrails, safety layers, structured prompting, or post, hoc validation have shown some reduction in risks but do not address the fundamental limitation that an LLM is intrinsically stochastic and generates outputs through probabilistic next, token prediction [23,30]. These added control mechanisms sit around the model rather than changing its underlying behaviour, meaning they can constrain or filter outputs but cannot guarantee determinism, faithfulness, or complete avoidance of hallucination. As more guardrails are layered on, the system becomes increasingly complex, harder to audit, and more difficult to reason about, while still lacking the reproducibility and explicit logic required in high, stakes decision, making.

Rule-based systems are designed for this setting. SESL is a domain, specific language and runtime for expressing decision logic as structured rules evaluated by a deterministic forward, chaining engine. SESL rules are written in a human, readable YAML, based format, providing clarity and auditability. The SESL engine evaluates rules deterministically, tracks dependencies, and produces detailed evaluation traces. SESL includes tooling for linting, scenario, based testing, and visual explanations such as driver trees.

This paper investigates whether a deterministic symbolic layer can mitigate reliability issues inherent to LLMs while preserving usability. It also explores if this approach overcomes the previous challenges of authoring the rules needed for rule-based systems.

We use SESL as a case study of a tool with a symbolic architecture intentionally designed to interoperate with LLMs. In this arrangement, LLMs assist with rule authoring, scenario generation, and explanatory text, whereas SESL retains responsibility for deterministic execution of all operational decisions.

This combination using neural networks (LLMs) for flexible interpretation and generation alongside symbolic systems (SESL) for reliable execution is known as a neuro, symbolic approach, and it aims to leverage the strengths of both paradigms while mitigating their respective weaknesses.

Contributions

1. We present SESL as a modern rule-based expert, system language focused on determinism and explainability
2. We propose a hybrid LLM–SESL architecture in which LLMs author and explain models while SESL executes decisions deterministically
3. We evaluate the hybrid approach on synthetic loan approval tasks, demonstrating perfect determinism and elimination of hallucinated justifications compared to LLM, only baselines

---

2. Problem Definition and Motivation

Organizations deploying automated decision, making in high, stakes commercial domains such as lending, underwriting, fraud detection, or compliance must satisfy:

1. Determinism: The same inputs must always produce the same outputs
2. Explainability: Every decision must include a traceable justification
3. Governance: Models must support review, change control, regression testing, and audit

LLMs alone fail these requirements due to hallucination [1–3], instability [4–6], and non, determinism. Research on hallucinations in LLMs identifies multiple contributing factors including training data quality, architectural design, and fundamental limitations of next, token prediction objectives. Recent work has shown that LLMs do not reliably use their intermediate chain-of-thought reasoning steps when generating answers, undermining the reliability of their explanations and causing failure to meet the governance criteria [4,5,15].

Traditional expert systems were deterministic and provide reliable outputs, but they have historically been costly to author manually and restrictive in scope. The effort required to encode domain knowledge into formal rules, maintain those rules as policies evolve, and handle edge cases has limited their practical adoption outside of mission, critical applications [39]. Additionally, classical expert systems often struggled with flexibility once the rules were written, adapting them to new scenarios or business contexts required substantial manual effort from domain experts.

Organizations need a hybrid architecture where decision logic remains symbolic and deterministic, but where the authoring and maintenance burden is significantly reduced. LLMs offer a potential solution to this historical bottleneck: they can rapidly translate natural language policies into structured rules, propose test scenarios, and assist with rule refinement—dramatically lowering the cost and expertise barrier that previously limited rule-based system adoption [11,12–14,19].

---

3. Background and Related Work

3.1 LLM Risks in High, Stakes Domains

In high, stakes settings such as lending, insurance underwriting, medical diagnosis, or regulatory compliance, a single incorrect decision (even if rare) can result in disastrous consequences, substantial financial loss, legal liability, reputational damage, and harm to individuals. Unlike other LLM actions such as web search or content generation where occasional errors may be tolerable, these domains require systems where unexpected or incorrect responses are minimized and, when they do occur, are fully traceable and explainable.

Beyond hallucination, LLMs present a fundamental opacity problem. These models are black, box systems with billions of parameters - it is impossible to understand precisely how, given a specific prompt, a particular response is actually produced. The internal computation involves complex, non, linear transformations across hundreds of layers of neural networks, making the path from input to output fundamentally untraceable.

With complex prompts and chains of prompts (as in multi, step reasoning or agentic workflows), this opacity compounds: each step's output becomes the next step's input, creating cascading dependencies that are impossible to fully validate.

While we can observe input, output behaviour and measure statistical patterns, we cannot prove that an LLM will respond correctly to novel inputs or edge cases. This makes validation extremely difficult and ultimately impossible to guarantee, rendering LLM, only systems unsuitable for high, stakes decisions where accountability and reproducibility are required.

3.2 Explainability and Governance

Regulators increasingly require transparency and traceability. The European Parliament AI Act [32], OECD [7], and NIST [8] frameworks require trustworthy, governable AI. There has already been much said about this need and how to approach compliance. Goodman and Flaxman's work on GDPR algorithmic accountability established the foundation for debates about rights to explanation in automated decision, making, while subsequent studies have clarified the scope and requirements of meaningful information about algorithmic logic, and Rudin argues for inherently interpretable models for safety, critical tasks, noting that post, hoc explanations for black, box models may perpetuate poor practices [11].

3.3 Symbolic and Neuro, Symbolic Systems

A symbolic system is an AI approach that represents knowledge explicitly through rules, logic, and structured facts, allowing it to reason deterministically and transparently. They provide stable, auditable decision logic.

In contrast, neural systems such as LLMs store knowledge implicitly in neural weights and generate outputs through probabilistic pattern prediction. LLMs provide broad linguistic and reasoning capabilities without guaranteed faithfulness or consistency.

Traditional rule engines such as Drools, Prolog, CLIPS, and modern Business Rules Management Systems (BRMS) also provide deterministic rule execution, but SESL differs in its fixed evaluation semantics, built, in explanation artifacts, and explicit design for LLM, assisted rule authoring and validation.

Fixed evaluation semantics means that SESL's rule engine follows a predetermined, invariant execution procedure: rules are always evaluated in the same order, conflicts are resolved using explicitly defined policies, and the same input facts always trigger the same sequence of rule firings. This contrasts with systems that may have configurable or implementation, dependent evaluation strategies, where subtle changes in configuration or runtime environment could alter execution behavior.

Fixed semantics ensure that SESL models are truly portable and reproducible - a model will behave identically across different deployments, time periods, and system versions.

SESL as a neuro-symbolic tool integrates across four features identified in recent surveys. It provides structured symbolic representations of tasks and states while incorporating neural components to learn patterns and latent properties from data. These learned elements enrich the symbolic structures and support SESL’s reasoning layer, which applies rules and constraints to produce transparent, explainable inferences. Finally, SESL combines the outputs of both learning and reasoning to guide decision-making, allowing it to select actions or evaluate plans in a way that reflects the complementary strengths of neural and symbolic approaches.

---

4. Hybrid LLM–SESL Method

4.1 Division of Responsibilities

In the Neuro-symbolic or hybrid LLM–SESL architecture, responsibilities are deliberately divided to balance flexibility with reliability. Large Language Models contribute interpretive and generative capabilities, translating natural, language policies into structured artifacts and improving human understanding of rule-based decisions. SESL, by contrast, serves as the deterministic execution and governance layer, ensuring that decisions are reproducible, auditable, and grounded in explicitly defined logic.

This separation prevents stochastic or opaque behavior from influencing outcomes while still leveraging LLMs' strengths in model authoring and explanation.

LLM Responsibilities:

- Generate initial rules from policy documents
- Refactor and document rules
- Propose test scenarios
- Produce natural, language explanations of rule execution

SESL Responsibilities:

- Provide structure and mechanisms for rules and scenarios
- Execute deterministic decisions
- Validate rules
- Provide traceable, grounded explanations

In this architecture, LLMs do not produce final decisions. They operate as authors, editors, and communicators. Decision authority is intentionally delegated to SESL to prevent stochastic variation, hallucinated justifications, or untraceable reasoning.

4.2 Why the Separation Matters

This clear division of responsibilities ensures trustworthiness, auditability, and operational safety:

- The LLM provides semantic richness, flexibility, translation of policy language, and improved usability for humans
- SESL provides the deterministic substrate, formal reasoning, strict validation, and reproducible explanations required for business, grade decisions

Together, they form a hybrid neuro, symbolic architecture [12–14] where creativity and interpretation stay on the LLM side, while correctness and accountability stay on the SESL side.

4.3 Cost and Performance Considerations

The development and management of complex LLM, based agentic systems entail substantial costs. Deploying these systems in high, volume or high, value transactional environments can become financially challenging. Moreover, the governance of such systems has proven challenging, often requiring specialized technical expertise.

Beyond direct operational costs, poorly implemented LLM, based solutions carry significant risk of failure and reputational damage. A prominent example is Air Canada's chatbot incident in 2024, where the airline's LLM, powered customer service assistant provided incorrect information about bereavement fare policies, leading to a legal ruling that held the company responsible for the chatbot's erroneous promises. This case illustrates how LLM hallucinations in customer, facing applications can result not only in individual customer harm but also in legal liability, negative press coverage, and erosion of trust. In regulated industries such as financial services, healthcare, or insurance, similar failures could trigger regulatory investigations, compliance violations, and substantial financial penalties in addition to reputational costs.

In contrast, rule-based systems are inherently low, cost to run. They operate with high speed, minimal computational resources, and straightforward infrastructure requirements, incurring negligible ongoing operational expenses. Historically, however, the primary limitation of rule-based approaches has been the significant effort and expense associated with authoring rules and constructing scenario logic. The major cost has been upfront: translating domain expertise into formal rules, maintaining those rules as policies evolve, and ensuring comprehensive coverage of business scenarios.

Airbus’s fly-by-wire flight-envelope protection system is a well-known rule-based safety system used on the A320 aircraft. It enforces deterministic rules that prevent pilots from exceeding certified aerodynamic and structural limits, overriding unsafe inputs when necessary.

An important advantage of rule-based systems is that, because they are deterministic, testing can be much more rigorous and comprehensive. Every decision pathway can be explicitly tested and confirmed through scenario, based test suites. Unlike probabilistic systems where exhaustive testing is impossible, deterministic rule systems allow organizations to validate that every branch of logic behaves correctly under all specified conditions. This testability dramatically reduces the risk of unexpected runtime behavior and provides strong assurance for regulatory compliance.

Although successful rule-based implementations exist, particularly in mission, critical contexts, the cost and complexity of the authoring process have inhibited broader, large, scale adoption [39]. Additionally, classical expert systems often struggled with flexibility - once rules were written, adapting them to new scenarios or business contexts required substantial manual effort from domain experts.

The emergence of LLMs has substantially reduced this barrier. By enabling the rapid generation of rules and scenarios through simple rule-based languages, LLMs significantly streamline the authoring process. This reduction in effort and cost substantially lowers the entry threshold for the adoption of rule-based systems, making it possible to combine low operational costs with reduced authoring costs.

---

5. SESL Architecture, Language, and Operational Model

SESL (Simple Expert System Language) is designed as a deterministic, explainable, and auditable rule-based decisioning framework suitable for high, stakes environments. It combines a human, readable rule language, a deterministic forward, chaining execution engine, and a suite of tooling for authoring, testing, and governance. Together, these components form a symbolic substrate that can be used independently or in hybrid workflows with LLMs.

5.1 Architecture Overview

SESL comprises three tightly integrated components:

1. A human, readable rule definition language, designed to be accessible to analysts yet precise enough for deterministic execution
2. A deterministic forward, chaining rule engine, which evaluates all rules until convergence under strict and configurable execution semantics
3. Tooling for development, testing, debugging, and explanation, including a rule linter, scenario runner, interactive execution shell, and dependency, graph generator

These components are built to support governance requirements—traceability, reproducibility, and auditability, so making SESL well, suited for regulated industries such as finance, insurance, healthcare, and taxation.

5.2 SESL Rule Language

The SESL language is a structured, YAML, like DSL incorporating three primary sections:

(a) Constants

A const block defines globally used values such as numeric thresholds, flags, or policy parameters. Constants ensure transparency and make policy updates safer by isolating configurable parameters.

(b) Rules

Each rule is a structured unit of decision logic with the following fields:

- rule: A unique, human, readable identifier
- priority (optional): Resolves conflicts when multiple rules write to the same target
- if: A boolean predicate over facts, constants, or computed values
- then: Assignments to result fields or derived facts
- reason: A human, readable explanation used in trace outputs

(c) Fact Scenarios

SESL includes a facts block representing test cases or input scenarios. Scenarios can be manually created, generated by LLMs, or part of an automated scenario test suite. Each scenario is a hierarchical structure reflecting input data (e.g., loan application, insured driver profile, transaction requiring VAT classification).

Language Features

The language supports:

- Numeric, boolean, and comparison operators
- Arithmetic via a strict LET expression subsystem
- Hierarchical dotted paths (e.g., applicant.credit.score)
- Explicit value assignment semantics
- Structured results under result.*

SESL enforces predictable evaluation semantics, making policies both transparent and auditable.

Example SESL Model

```
model: Fraud Detection
meta: {}

const:
  supply_value_limit: 5000

rules:
```

```
, rule: HighValueConsumerFlag
 if:
  all:
  , customer.type == "consumer"
  , supply.value >= supply_value_limit
 then:
  result.flag_high_value_consumer: true

, rule: CountryMismatchFlag
 if: supplier.country != customer.country
 then:
  result.flag_country_mismatch: true

facts:
, scenario: Fraud Example
 supply:
  id: F1
  type: service
  category: general
  value: 4000
 supplier:
  country: SG
  vat_registered: false
 customer:
  type: consumer
  country: AU
  vat_registered: false
 result: {}
```

5.3 Deterministic Forward, Chaining Engine

The SESL engine interprets the rule model using a deterministic forward, chaining algorithm designed for clarity, reproducibility, and robustness. Forward, chaining means the engine starts with known facts and works forward through the rules, applying each rule whose conditions are met and updating the working state with new conclusions. This continues in cycles until no more rules can fire, a state called convergence or fixed point.

The engine follows a straightforward execution process. First, it loads and validates the model, parsing all rules, constants, and fact scenarios while catching any errors such as invalid identifiers, malformed expressions, or missing data paths. Then it begins iterative rule evaluation, examining each rule in sequence. For every rule, one of three things happens: the rule matches (its condition evaluates to true and its actions are applied), the rule fails (its condition evaluates to false and nothing happens), or the rule errors (due to an invalid expression, missing data, or unsafe reference).

When a rule fires, its actions modify the working fact state, perhaps setting a result value or creating a derived fact. The engine then starts another evaluation cycle, checking all rules again with this updated state. This process repeats until the system reaches a fixed point where no rule can change the state further, or until a rule explicitly stops execution, or until a safety iteration limit is reached to prevent infinite loops.

Understanding Deterministic Conflict Handling

In any rule-based system, conflicts can arise when multiple rules want to write to the same result field. For example, one rule might say "set credit_decision to APPROVED" while another says "set credit_decision to DECLINED." SESL handles these conflicts deterministically, meaning the resolution is predictable and consistent across all executions.

SESL supports configurable conflict resolution policies. One common approach is priority, based resolution: each rule can have a priority number, and higher priority rules override lower priority ones. Another approach is order, based resolution: the last rule to fire wins. Crucially, these policies are explicit and deterministic - identical inputs always produce identical outputs, regardless of when or where the model runs. This differs from some systems where conflict resolution might depend on the order rules happen to be stored in memory or other implementation details that could vary between runs.

Trace and Dependency Recording

Each rule evaluation is logged in detail, creating an audit trail for debugging and explanation. The trace records which rules were evaluated, which fired, which failed, what values were computed, and how the state changed at each step. This comprehensive logging is what makes SESL's explanations trustworthy they're derived directly from the actual execution path, not generated after the fact.

Strict Mode Safety Features

SESL's strict execution modes ensure reliability by catching potential problems before they cause issues. Unknown identifiers produce immediate errors rather than silently defaulting to null or empty values. Unquoted text cannot silently become a string, everything must be explicit. Unsafe constructs such as arbitrary function calls or external references are rejected outright. The LET expressions (for computing derived values) are evaluated through a restricted interpreter that prevents arbitrary code execution.

These safety features guarantee that SESL models cannot behave unpredictably, even if they were authored or modified by LLMs without perfect understanding of the domain. The system enforces correctness structurally, making it much harder to accidentally create rules that produce unexpected behavior.

5.4 Monitoring, Tracing, and Explanation Framework

A defining strength of SESL is its built, in explanation layer. Every evaluation emits a comprehensive structured trace that tells the complete story of how a decision was reached. The explanation system captures several types of information. Rule firing traces provide a chronologically ordered list of everything that happened during evaluation: which rules were examined, which ones fired, which failed and why, and which conditions were met or unmet at each step. This creates a complete audit trail of the decision process.

Driver trees take this a step further by mapping the causal relationships in the decision. They show how input facts led to final outputs, which rules contributed to each result value, and the dependencies between facts, conditions, and outcomes. Think of a driver tree as a visual map showing "this credit score led to this rule firing, which set this flag, which caused this final decision." These visualizations are especially valuable for regulatory inspection, internal audit, and explaining decisions to customers who want to understand why they received a particular outcome.

Explanation blocks provide human, readable summaries that translate the technical execution into natural language. They describe which conditions were matched, what values were computed in LET expressions, what result values were assigned, and why - all drawn from the reasons specified in rule definitions. This makes explanations accessible to non, technical stakeholders.

Finally, execution metrics provide quantitative information about the evaluation: how many iterations were needed, how many rules matched and fired, performance timings, and whether any conflict resolution events occurred. Together, these artifacts enable step, by, step reconstruction of the decision process, meeting governance obligations for transparency. More importantly, they're faithful to what actually happened.

5.5 Tooling and Development Workflow

The SESL CLI (command, line interface) provides a professional, grade environment for model development and governance, making it practical to build and maintain complex rule sets in real business contexts.

The interactive execution mode is particularly valuable during development. It enables step, by, step evaluation of rules, letting developers and domain experts watch exactly how the engine processes each rule, what values change, and why. This is invaluable for debugging complex logic and for training new team members on how policies work in practice.

For production use, the batch execution mode supports CI/CD pipelines, regression testing, and bulk scenario runs. Organizations can integrate SESL into their automated testing infrastructure, running thousands of test scenarios on every rule change to ensure nothing breaks. This level of automated validation is difficult to achieve with probabilistic systems but straightforward with deterministic rule engines.

The linting and structural validation tools act as a safety net, catching problems before they reach production. The linter detects issues like unreachable rules that can never fire, unused constants that suggest incomplete implementation, conflicting assignments where multiple rules fight over the same value, missing fact paths that would cause runtime errors, and other unintended model behaviors. Running the linter before deployment catches most common authoring errors.

The scenario testing framework allows teams to build large suites of test inputs that validate policy behavior across edge cases. Because SESL is deterministic, these scenarios function as executable specifications—if a scenario passes, you know that particular input will always produce that output. This makes scenario libraries a powerful tool for regression testing and policy drift detection.

Finally, the dependency graph generation produces visualizations that show the structure of the rule model: which rules depend on which facts, how information flows from inputs to results, and where circular or redundant logic might exist. These graphs help teams understand complex models at a glance and identify opportunities for simplification or refactoring.

5.6 Expert Review and Governance

SESL fits naturally into business governance workflows because it bridges the gap between technical implementation and business policy. The human, readable rule format means business experts can review rule text directly, without needing to understand programming languages or reverse, engineer code. They can verify that the implemented rules actually match the intended policy, catch logical errors, and suggest improvements.

Engineers, on the other hand, validate execution traces for technical correctness. They ensure the rules execute efficiently, handle edge cases properly, and integrate correctly with surrounding systems. The clear separation between business logic (in rules) and technical implementation (in the engine) makes this division of responsibility practical.

Risk and compliance teams audit models using SESL's deterministic logs. Because every decision is accompanied by a complete trace showing exactly which rules fired and why, auditors can verify that the system is following approved policies. They can also investigate specific decisions retrospectively—if a customer complains or a regulator asks questions, the organization can produce a detailed explanation of how that decision was made, which policy provisions were applied, and what data was used.

SESL includes built, in descriptive metadata such as data sources, creation and update dates, owner information, and other documentation fields. Combined with version control, this ensures traceability of every rule modification. Organizations can track who changed what, when, and why. This is essential for regulated environments where changes must be documented and justified.

In this governance model, LLMs may assist in drafting or refactoring rules, proposing new scenarios, or generating documentation. However, SESL remains the authoritative execution environment. LLM suggestions go through the same review and testing process as manually authored rules. This makes SESL suitable for large, regulated organizations that require demonstrably transparent and reproducible decision, making systems while still benefiting from LLM acceleration in the authoring phase.

---

6. Evaluation of the Hybrid LLM–SESL Method

6.1 Experimental Setup

To evaluate the benefits of symbolic execution relative to purely generative reasoning, we compared two system configurations:

(a) LLM, only system

In this baseline condition, a Large Language Model (GPT, 4o with temperature 0.0) receives both the natural, language policy description and a structured case profile. The LLM is asked to determine the correct decision and provide a justification. All reasoning and explanation are generated internally by the model. This condition represents the common industry pattern of using an LLM directly as a decision engine.

(b) LLM, generated SESL rules executed by SESL

In this condition, the LLM is used only during model authoring: it translates the natural, language policy into a SESL rule model. After rule creation, SESL evaluates each case deterministically using its forward, chaining rule engine. Explanations are produced procedurally from SESL's rule, firing trace.

6.2 Experimental Task

We evaluated system performance on a loan eligibility assessment task. This task models a lending policy that incorporates income thresholds, debt, to, income ratios, credit scores, and employment stability. Each scenario represents a synthetic loan application to be evaluated.

The typical business process (based on common regulatory lending workflows, e.g., UK FCA Responsible Lending Guidelines [20]) includes:

1. Collect applicant information: income, debts, credit history, employment
2. Validate submitted documents: pay slips, bank statements, ID verification

3. Assess affordability: compute debt, to, income and disposable income
4. Evaluate creditworthiness: retrieve credit bureau metrics
5. Apply policy rules: check thresholds, cutoffs, and exceptions
6. Determine outcome: approve, require manual review, or decline
7. Generate rationale: documented reasons for audit and customer communication

Method

We generated 100 synthetic loan applications with realistic but non, sensitive distributions for all input features. A policy document was created specifying three, tier decision logic (APPROVED / DECLINED / MANUAL REVIEW). Before conducting any experiments, all 100 applications were manually assessed by domain experts to create an independent ground truth benchmark against which both systems could be evaluated.

For the LLM, only baseline, we used GPT, 4o (temperature 0.0, JSON response format) with prompts asking for a decision and justification. For the SESL condition, we used an LLM to generate SESL rules from the policy document, then executed each application through SESL's deterministic engine.

We conducted six independent runs (run 0 through run 5) for each system. Between run 0 and run 1, and again between run 4 and run 5, we applied prompt, tuning to the GPT, based system to improve performance based on error analysis. The prompts and full methodology are documented in Appendix A.

6.3 Results

Decision Distribution Accuracy

Table 1 shows the distribution of decisions produced by GPT, SESL, and the manually validated benchmark across all 600 evaluations (100 scenarios × 6 runs).

Table 1: Distribution of Decisions

Decision	GPT Count	SESL Count	Manual Count	GPT %	SESL %	Manual %
APPROVED	177	138	138	30%	23%	23%
DECLINED	256	282	282	43%	47%	47%
MANUAL REVIEW	167	180	180	28%	30%	30%
Total	600	600	600	100%	100%	100%

SESL perfectly reproduced the manually validated decision distribution. This result is expected: SESL is a deterministic rule-based system, and provided the rules are correctly authored it will always return the same output for a given scenario. In this study, the underlying process was captured accurately within the SESL rule set; the use of the LLM to assist in drafting SESL rules helped overcome the historically significant barrier associated with manual rule authoring.

GPT, by contrast, showed both distributional divergence and individual decision errors. Across the 600 total evaluations (100 scenarios × 6 runs), GPT's aggregate distribution deviated from the manual benchmark in its total counts. When examining the distribution across all runs, GPT produced 39 more APPROVED decisions and 26 fewer DECLINED decisions than the manual benchmark, while MANUAL REVIEW counts were closer but still off by 13.

To test whether GPT's overall decision distribution differs significantly from the manual distribution, we performed a chi, square goodness, of, fit test comparing the aggregate counts across all 600 evaluations. The test yielded $\chi^2 = 14.36$, $p = 0.00076$, indicating that GPT's distribution is statistically significantly different from the manual benchmark. This suggests systematic bias in how GPT applies the policy rules—it tends to approve more cases and decline fewer cases than the policy actually requires.

However, the more critical measure for operational use is individual decision accuracy. Examining the raw evaluation data reveals that GPT made incorrect decisions on specific applicants, though the exact error rate varies across runs due to the stochastic nature of LLM outputs. The distributional mismatch of 78 total decisions (39 + 26 + 13) across 600 evaluations represents approximately 13% aggregate deviation, though this does not directly translate to a 13% per, scenario error rate due to the repeated, measures design.

Individual Decision Accuracy Analysis

To assess per, scenario accuracy, we analyzed run 0 (the baseline run before any prompt tuning) in detail. In this run, GPT made incorrect decisions on 42 out of 100 scenarios (42% error rate) when compared to the manual ground truth. After prompt engineering improvements implemented between run 0 and run 1, the error rate decreased substantially to approximately 22%, though volatility remained an issue as discussed below.

Determinism Analysis

Table 2 presents the distribution of decisions across six repeated runs for both systems.

Table 2: Distribution of Decisions by Run

GPT Runs:

Decision	run 0	run 1	run 2	run 3	run 4	run 5	Total
APPROVED	42	29	29	26	27	24	177
DECLINED	36	43	46	46	46	39	256
MANUAL REVIEW	22	28	25	28	27	37	167
Total	100	100	100	100	100	100	600

SESL Runs:

Decision	run 0	run 1	run 2	run 3	run 4	run 5	Total
APPROVED	23	23	23	23	23	23	138
DECLINED	47	47	47	47	47	47	282
MANUAL REVIEW	30	30	30	30	30	30	180
Total	100	100	100	100	100	100	600

As expected for a deterministic rule-based system, SESL produced identical outputs across all runs, yielding perfect consistency for each decision category.

The GPT, based system exhibited run, to, run variability, even though input data remained unchanged in runs 0–4. The differences across GPT runs were not statistically significant ($\chi^2 = 13.66$, $p = 0.189$), indicating fluctuations within the range expected from probabilistic sampling. However, the magnitude and direction of changes between specific runs correspond to the points at which prompt adjustments were introduced, demonstrating the sensitivity of LLM outputs to prompt design.

Volatility Analysis

To assess GPT's decision stability, we analyzed how often an applicant received different outcomes across runs 1–4 (during which neither prompt nor input data were modified). Eight applicants (out of 100) exhibited at least one change in decision across these runs, with fluctuations distributed as follows: 5 changes in run 1, 4 in run 2, 2 in run 3, and 3 in run 4.

Table 3: Complete GPT Decision Inconsistencies Across Controlled Runs

Applicant ID	Decision	run 0	run 1	run 2	run 3	run 4	run 5
8	APPROVED	1	1	1	—	—	—

Applicant ID	Decision	run 0	run 1	run 2	run 3	run 4	run 5
	MANUAL REVIEW	—	—	—	1	1	1
16	MANUAL REVIEW	1	1	—	—	—	1
	DECLINED	—	—	1	1	1	—
18	APPROVED	1	1	—	—	—	—
	MANUAL REVIEW	—	—	1	1	1	1
23	DECLINED	1	—	1	1	1	1
	MANUAL REVIEW	—	1	—	—	—	—
51	APPROVED	1	—	1	—	—	—
	MANUAL REVIEW	—	1	—	1	1	1
55	APPROVED	1	1	1	—	1	—
	DECLINED	—	—	—	1	—	1
64	MANUAL REVIEW	1	1	—	—	—	—
	DECLINED	—	—	1	1	1	1
66	DECLINED	1	—	1	1	1	1
	MANUAL REVIEW	—	1	—	—	—	—
94	DECLINED	—	1	1	—	1	—
	MANUAL REVIEW	1	—	—	1	—	1

Note: "1" indicates the decision was rendered for that applicant in that run; "—" indicates no decision was made. Each applicant showed at least one decision change across runs 1, 4 (the controlled period with identical prompts and inputs).

This yields an approximate volatility rate of 3–5% across controlled conditions, demonstrating that GPT introduces measurable non, determinism even when all experimental conditions remain fixed. For example, Applicant 8 received APPROVED in runs 0, 1, and 2, but then switched to MANUAL REVIEW in runs 3, 4, and 5—despite identical inputs. Applicant 55 alternated between APPROVED and DECLINED across multiple runs. Applicant 94 showed particularly erratic behaviour, switching between

DECLINED and MANUAL REVIEW across different runs with no clear pattern. This volatility represents a fundamental reliability concern for operational deployment, as the same loan application could receive different outcomes depending on when it happens to be processed.

---

7. Discussion

Our evaluation highlights complementary strengths across symbolic and LLM approaches and demonstrates that neither alone is sufficient for all aspects of high, stakes decision automation.

LLMs provide powerful capabilities for interpreting policy text, generating candidate rule structures, and supporting human understanding, but their stochastic behavior and variable reasoning limit their suitability as standalone decision engines. Symbolic systems such as SESL, by contrast, offer determinism, traceability, and governed execution, though they require more effort to author and maintain. Together, the two approaches form a hybrid architecture that balances flexibility with reliability.

7.1 Key Insights

Viewed together, the results show how LLMs and symbolic systems occupy different but complementary roles in a high, stakes decision pipeline. The key findings can be summarized as follows:

1. Determinism requires symbolic execution

Determinism is not an incidental property: it is a direct consequence of symbolic execution. SESL's forward, chaining engine, fixed evaluation semantics, and explicit conflict resolution policies ensure that the same inputs always lead to the same outputs, across runs and environments. In domains where organizations must justify and reproduce historical decisions (often years after the fact) this kind of stability is not merely convenient but essential.

2. LLMs exhibit systematic bias and volatility despite careful configuration

The GPT, based system illustrated how even a carefully configured LLM can drift in subtle ways. Despite using a fixed model, temperature 0.0, and consistent inputs, we observed decision volatility across runs (3, 5%), along with systematic skew in aggregate decision distributions relative to the policy, defined benchmark. Prompt engineering reduced (but did not eliminate) these issues, improving accuracy from 42% error rate to 22% but still leaving substantial reliability concerns. This is not a criticism of the model as such; it is a reflection of the underlying generative paradigm. When a system computes answers via probabilistic token prediction, small shifts in prompting, context, or model behavior can manifest as different outcomes, especially near decision boundaries.

3. Faithful explanations derive from execution traces, not post, hoc generation

The study underscores a key distinction between what a system decides and how it explains that decision. SESL's explanations are produced from the actual execution trace: they derive from the concrete sequence of rules that fired and the values that were set. As a result, SESL's explanations are faithful by construction. GPT, in contrast, produces justifications post, hoc, in natural language, and the literature suggests that such chain-of-thought style explanations need not match the model's true internal reasoning. In our experiments, GPT sometimes offered plausible narratives that masked misapplications of policy thresholds or compensatory reasoning that the policy did not permit.

4. LLMs excel as authoring tools, not decision engines

The evaluation shows that LLMs are extremely effective before deployment as tools for authoring, refactoring, and documenting rule sets. The SESL model used in the experiments originated from LLM, assisted translation of a policy document into SESL rules. Without that assistance, the effort required to build and iterate a comparable rule model would have been substantially higher. In this sense, LLMs help address the classic authoring bottleneck that has historically limited rule-based systems.

5. Division of labor enables safe use of LLMs in high, stakes contexts

These observations support a division of labor: LLMs serve best as policy interpreters, draft rule writers, and explanation generators conditioned on symbolic traces, while SESL functions as the authoritative decision engine. The hybrid arrangement does not ask LLMs to provide guarantees they cannot meet; instead, it constrains their influence to stages where errors are detectable through testing and review. Symbolic execution, in turn, provides the deterministic substrate and faithful explanations that regulators, auditors, and affected individuals increasingly expect.

7.2 Limitations

Real, world models may scale to thousands of rules

In large organizations, operational decision engines often encode decades of policy evolution, regulatory constraints, product variations, and exception handling. A mature deployment may require managing thousands of rules across multiple interacting models. While SESL's structured language, linter, and scenario, based testing are designed to support this scale, the cognitive and organizational complexity of maintaining extensive rule bases cannot be eliminated entirely [37,38].

Ensuring consistency across interconnected rule sets, preventing logic duplication, and managing change control remain significant engineering challenges. Future development of higher, level abstraction mechanisms, modularization frameworks, and automated rule analysis tools will be essential to support the long, term sustainability of SESL in business environments.

LLM prompting requires governance

Although LLMs can accelerate rule model creation by translating natural language policies into candidate rule structures, their outputs remain sensitive to prompt phrasing, model version, and context. Without guardrails, an LLM may introduce subtle misinterpretations of policy language or fail to capture critical edge conditions.

LLM use in SESL workflows must operate within a formal governance framework that includes versioned prompts, validation checkpoints, reproducibility controls, and mandatory human review. Organizations should treat LLM, generated rules as suggestions rather than executable policy until they have passed automated linting, regression testing, and expert verification.

Human review remains essential

Despite SESL's deterministic semantics and comprehensive explanation capabilities, the correctness of a SESL model ultimately depends on the human experts who define and validate the underlying policies. Automated evaluation can detect structural issues in rules, but it cannot determine whether the encoded policy is itself fair, lawful, or accurate.

Domain experts must review proposed rule changes, validate scenario outcomes, and assess alignment with business intent and regulatory requirements. Furthermore, because LLM, generated rules are not inherently trustworthy, expert oversight is critical to ensure that natural language ambiguity does not propagate into the decision logic.

Business integration complexity

Deploying SESL in production requires integration with existing business systems such as loan origination platforms, underwriting engines, CRM systems, and compliance monitoring workflows. These integrations must accommodate data ingestion pipelines, identity and access management, monitoring infrastructure, and version, controlled deployment environments.

Additionally, organizations may need to align decision outputs with downstream audit, case, management, and reporting systems to satisfy internal and external regulatory requirements. Ensuring seamless bidirectional interaction between SESL, LLM services, and operational databases can be non, trivial, particularly in environments with strict security and governance constraints.

The rule authoring bottleneck

While SESL provides deterministic execution, the cost of rule authoring remains substantial. Domain experts are needed to author and validate rule sets for moderately complex policies. This can represent a significant upfront investment compared to supervised ML approaches, which may achieve comparable accuracy with labeled data alone. Organizations must weigh this authoring cost against the benefits of interpretability and governance.

LLM, generated rules are not automatically correct

A critical tension exists in the hybrid approach: we argue LLMs cannot be trusted for decision execution due to hallucination, yet we propose using LLMs for rule authoring. This is not a contradiction but rather a risk, reduction strategy. LLM errors in rule authoring are detectable through validation, testing, and expert review before deployment, whereas LLM errors in runtime decision, making are not. However, this does not eliminate the risk of LLM, introduced policy misinterpretations, and organizations must implement rigorous validation workflows.

Comparison with modern interpretable ML

Our evaluation does not include comparisons with other interpretable ML approaches such as gradient boosted decision trees (XGBoost, LightGBM), Generalized Additive Models (GAMs), or modern rule, learning approaches (RuleFit, Skope, rules). These methods offer different tradeoffs between interpretability, performance, and governance requirements.

SESL is most appropriate when: (1) explicit policy encoding is required by regulation, (2) decisions must be reproducible across system versions, or (3) domain experts require direct control over decision logic. For applications where learned patterns are acceptable, interpretable ML may be more cost, effective.

7.3 Broader Impacts

The adoption of SESL as a symbolic substrate beneath LLM interfaces has the potential to significantly improve the transparency, accountability, and trustworthiness of AI systems used in socially consequential domains. By ensuring that all operational decisions are computed deterministically and accompanied by faithful, structurally grounded explanations, SESL reduces the risk of opaque model behavior and enables more robust oversight by auditors, regulators, and internal governance teams.

This stands in contrast to existing black, box or LLM, only systems, which may obscure sources of error, embed hidden biases, or provide explanations that diverge from actual reasoning processes. SESL therefore acts not only as a technical reliability layer but also as an institutional governance aid, making complex AI systems more accessible to non, technical stakeholders involved in compliance, policy review, and risk management.

However, SESL does not eliminate broader ethical risks inherent to automated decision, making. The accuracy and fairness of the system ultimately depend on the quality of the policies, data, and domain knowledge encoded as rules, and these may reflect historical inequities or incorrect assumptions. While SESL increases the inspectability of such issues, it cannot resolve them on its own; responsible deployment requires rigorous fairness analysis, legal review, and continuous monitoring for disparate impact.

Furthermore, the use of LLMs in rule authoring introduces new challenges, including the potential importation of societal biases or misinterpretation of policy language. Although SESL's deterministic execution prevents these issues from directly appearing in operational decisions, organizations must employ appropriate validation workflows to ensure that LLM, generated rules align with domain standards and ethical norms.

7.4 Decision Framework: When to Use SESL vs. Alternatives

SESL is not universally optimal. We recommend SESL when:

Strong Indicators:

- Regulatory requirements mandate explicit, auditable decision logic
- Policies change frequently and must be traceable to business decisions
- Domain experts must directly control decision rules
- Decisions must be reproducible across system versions for compliance
- Post, hoc explanations are insufficient (pre, hoc interpretability required)

Indicators Against:

- Optimal patterns are unknown and must be learned from data
- Decision boundaries are highly complex and non, linear
- Authoring cost exceeds model training and validation cost
- Domain expertise is limited or unavailable
- Rapid prototyping and iteration are priorities

Alternative Approaches to Consider:

- Supervised ML with interpretability constraints: When patterns must be learned but some interpretability is needed (use constrained trees, GAMs, RuleFit)
- Traditional ML + explanation layer: When black, box performance is acceptable and post, hoc explanations suffice (use XGBoost + SHAP)
- Hybrid symbolic, ML: When some rules are explicit and others learned (use rule-based preprocessing + ML)

7.5 Ethical Considerations

The use of SESL has the potential to bring several positive ethical benefits. By making automated decision-making processes more transparent, SESL can help reduce the risk of harm caused by opaque algorithms and improve regulatory compliance. This increased clarity can also strengthen public trust and support the preservation of institutional knowledge in a form that remains interpretable and auditable over time.

However, SESL also carries important risks that must be acknowledged. Transparency can create a false sense of security, as interpretable systems are not inherently fair or correct. Organizations might also engage in regulatory arbitrage, using SESL to give the appearance of compliance while leaving core decision-making processes opaque. Furthermore, explicit rules can encode bias just as easily as machine-learning models, and reliance on rule-based systems may discourage innovation by limiting exploration of new or unconventional decision patterns.

To mitigate these risks, SESL should be deployed alongside (rather than instead of) rigorous fairness auditing. Organizations must continue to test for disparate impact even when decisions are interpretable, and expert reviews should incorporate diverse perspectives rather than focusing solely on technical validation. Continuous monitoring is essential to detect unintended consequences that may emerge over time.

There are also dual-use concerns. Although SESL is intended to support responsible governance, it could be misused to automate discriminatory decisions under the guise of transparency, to encode questionable policies in an interpretable format, or to technically comply with oversight requirements while undermining their intent. For these reasons, we advocate deploying SESL only within comprehensive AI governance frameworks that prioritize fairness, accountability, and meaningful human oversight.

---

8. Conclusion

SESL demonstrates that deterministic symbolic execution can address specific limitations of LLM, only approaches in high, stakes business AI - particularly hallucination, non, determinism, and ungovernable explanations.

Our evaluation shows that Neuro-Symbolic / hybrid LLM–SESL architectures achieve perfect determinism and eliminate fabricated justifications, though at the cost of increased authoring effort and reduced flexibility compared to learned models.

SESL is not a universal solution. It is most appropriate for regulated domains where explicit policy encoding, direct expert control, and pre, hoc interpretability are required or strongly preferred. Organizations should evaluate SESL against modern interpretable

ML alternatives (gradient boosted trees, GAMs, rule, learning systems) based on their specific governance requirements, available expertise, and cost constraints.

Key Contributions

1. A modern expert system language with business tooling
2. Demonstration that symbolic substrates can mitigate LLM risks
3. A validation framework for LLM, generated rules
4. Empirical evidence on the determinism–flexibility tradeoff

Future Work

Future research should focus on:

- Real, world deployment studies with cost–benefit analysis
- Formal verification methods for rule sets
- Automated bias detection and remediation tooling
- Hybrid approaches combining SESL with learned components

We believe SESL represents a practical path toward trustworthy AI for specific business contexts, but acknowledge it is one tool among many in the broader interpretable AI landscape.

---

9. Data and Code Availability

Synthetic Test Data: The 100 synthetic scenarios used in our evaluation are available at www.sesl.ai/paper or upon request to reviewers. These scenarios contain no sensitive information and can be freely reproduced.

SESL Engine: The SESL runtime implementation is made available for research purposes under a non, commercial license at https://www.sesl.ai. Academic researchers may request full source code access for independent verification.

Evaluation Scripts: Python scripts for computing evaluation metrics (determinism, accuracy, stability analysis) are available at www.sesl.ai/paper or in supplementary materials.

LLM Prompts: All prompts used for LLM, only baseline and rule generation are documented in Appendix A to enable replication.

Reproducibility Note: LLM outputs may vary across API versions even with fixed parameters. Our evaluation used GPT, 4o with temperature 0.0. Independent researchers attempting replication should expect minor numerical variation but qualitatively similar results.

References

[1] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, 55(12), 1, 38.

[2] Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., & Liu, T. (2023). A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. arXiv preprint arXiv:2311.05232.

[3] Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y., Wang, L., Luu, A. T., Bi, W., Shi, F., & Shi, S. (2023). Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. arXiv preprint arXiv:2309.01219.

[4] Turpin, M., Michael, J., Perez, E., & Bowman, S. R. (2024). Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-thought Prompting. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*.

[5] Lanham, T., Chen, A., Radhakrishnan, A., Steiner, B., Denison, C., Hernandez, D., Li, D., Durmus, E., Hubinger, E., Kernion, J., Lukošiūtė, K., Nguyen, K., Cheng, N., Joseph, N., Schiefer, N., Rausch, O., McCandlish, S., Kundu, S., Kadavath, S., ... Perez, E. (2023). Measuring Faithfulness in Chain-of-thought Reasoning. arXiv preprint arXiv:2307.13702.

[6] Wiegreffe, S., & Marasović, A. (2021). Teach Me to Explain: A Review of Datasets for Explainable Natural Language Processing. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.

[7] OECD. (2019). Recommendation of the Council on Artificial Intelligence, OECD/LEGAL/0449. *OECD Legal Instruments*. https://legalinstruments.oecd.org/en/instruments/OECD, LEGAL, 0449

[8] National Institute of Standards and Technology (NIST). (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. U.S. Department of Commerce.

[9] Goodman, B., & Flaxman, S. (2017). European Union Regulations on Algorithmic Decision, Making and a "Right to Explanation". *AI Magazine*, 38(3), 50, 57.

[10] Wachter, S., Mittelstadt, B., & Floridi, L. (2017). Why a Right to Explanation of Automated Decision, Making Does Not Exist in the General Data Protection Regulation. *International Data Privacy Law*, 7(2), 76, 99.

[11] Rudin, C. (2019). Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence*, 1(5), 206, 215.

[12] De Raedt, L., Dumančić, S., Manhaeve, R., & Marra, G. (2020). From Statistical Relational to Neuro, Symbolic Artificial Intelligence. In *Proceedings of the Twenty, Ninth International Joint Conference on Artificial Intelligence (IJCAI, 20)* (pp. 4943, 4950).

[13] Besold, T. R., d'Avila Garcez, A., Bader, S., Bowman, H., Domingos, P., Hitzler, P., Kühnberger, K., U., Lamb, L. C., Lowd, D., Lima, P. M. V., de Penning, L., Pinkas, G., Poon, H., & Zaverucha, G. (2017). Neural, Symbolic Learning and Reasoning: A Survey and Interpretation. arXiv preprint arXiv:1711.03902.

[14] Garcez, A. d., Gori, M., Lamb, L. C., Serafini, L., Spranger, M., & Tran, S. N. (2019). Neural, Symbolic Computing: An Effective Methodology for Principled Integration of Machine Learning and Reasoning. *Journal of Applied Logics*, 6(4), 611, 632.

[15] Chen, Y., Zhong, R., Zha, S., Karypis, G., & He, H. (2022). Meta, Learning via Language Model In, context Tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 719, 730).

[16] Kline, D. M., & Walters, D. J. (2021). Machine Learning in Credit Risk Modeling: Efficiency Comes at a Cost. *The Journal of Risk Model Validation*, 15(1), 91, 110.

[17] Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015). Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30, day Readmission. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1721, 1730).

[18] Selbst, A. D., & Powles, J. (2017). Meaningful Information and the Right to Explanation. *International Data Privacy Law*, 7(4), 233, 242.

[19] Jackson, P. (1998). *Introduction to Expert Systems* (3rd ed.). Addison, Wesley.

[20] Financial Conduct Authority (FCA). (2023). *Consumer Credit Sourcebook (CONC)*. FCA Handbook. https://www.handbook.fca.org.uk/handbook/CONC/

[21] OECD. (2017). *International VAT/GST Guidelines*. OECD Publishing, Paris. https://doi.org/10.1787/9789264271401, en

[22] Marcus, G., & Davis, E. (2020). *Rebooting AI: Building Artificial Intelligence We Can Trust*. Pantheon Books.

[23] Chao, P., Robey, A., Dobriban, E., Hassani, H., Pappas, G. J., & Wong, E. (2023). Jailbreaking Black Box Large Language Models in Twenty Queries. arXiv preprint arXiv:2310.08419.

[30] Lipton, Z. C. (2018). The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability is Both Important and Slippery. *Queue*, 16(3), 31, 57.

[31] Park et al. (2023). Generative Agents: Interactive Simulacra of Human Behaviour. *UIST*.

[32] European Parliament. (2024). Regulation (EU) 2024/1689 on Artificial Intelligence (AI Act). *Official Journal of the European Union*.

[37] Brachman, R. J., & Levesque, H. J. (2004). *Knowledge Representation and Reasoning*. Morgan Kaufmann.

[38] Vanthienen, J., & Mues, C. (2006). Audit and Verification of Business Rule Models. *Expert Systems with Applications*, 30(4), 570–582.

[39] Hayes, Roth, F. (1985). Rule-based systems. *Communications of the ACM*, 28(9), 921–932.

Appendix A. Prompts, Programs, and Data Description

A.1 Model and Methodology for Prompt Improvement

Model: GPT, 4o (OpenAI)
Temperature: 0.0 (deterministic mode)
Response Format: JSON object (enforced via OpenAI API parameter)

Prompt engineering followed an iterative, data, driven approach:

1. Baseline Evaluation: Run initial prompt, measure agreement with SESL ground truth
2. Disagreement Analysis: Categorize all GPT, SESL mismatches by decision type and reason
3. Pattern Identification: Identify systematic errors (e.g., auto, approving manual review cases)
4. Hypothesis Formation: Determine root causes (e.g., LLM applying compensatory logic)
5. Prompt Refinement: Add explicit prohibitions and structural constraints
6. Validation: Re, run evaluation, measure improvement, iterate

A.2 Prompt Evolution for GPT Response

Initial Prompt (run 0 , Baseline)

You are a credit underwriter. The policy defines THREE decision categories:

APPROVED/DECLINED/MANUAL REVIEW.

Instructions: Follow policy thresholds precisely, return JSON with decision and reason.

Structure: Minimal, generic underwriter framing.

Result: 42 APPROVED, 36 DECLINED, 22 MANUAL REVIEW (58% agreement)

Observed Issues (from disagreement analysis):

- Auto, approved 19 cases that required MANUAL REVIEW (credit 640, 699)
- Applied compensatory logic ("high income compensates for borderline credit")
- Treated MANUAL REVIEW as "suggested human involvement" not mandatory decision
- Missed document verification failures
- Aggregated multiple positive factors to override single policy violation

Prompt Engineering Iteration 1 (runs 1, 4)

Added structured sections to enforce policy adherence:

CRITICAL INSTRUCTIONS:

1. THREE distinct decision categories (APPROVED/DECLINED/MANUAL REVIEW)

2. MANUAL REVIEW is NOT optional , must return if policy specifies

3. Apply ALL thresholds as exact cutoffs, not guidelines

4. Process rules in order, most restrictive takes precedence

PROHIBITED BEHAVIORS:

, Do NOT apply judgment to "borderline" cases

, Do NOT auto, approve manual review cases

, Do NOT make assumptions about unstated thresholds

, Do NOT aggregate factors to override explicit rules

Result: 27, 29 APPROVED, 43, 46 DECLINED, 25, 28 MANUAL REVIEW (70, 74% agreement)

Remaining Issues:

- Still auto, approving 6 manual review cases (credit score 640, 699)
- Too lenient on 10 document verification failures
- Misinterpreting credit 620, 639 range
- No explicit rule execution order

Prompt Engineering Iteration 2 (run 5)

Added rule precedence and absolute prohibitions:

RULE EXECUTION ORDER (MANDATORY):

1. Document Verification (FIRST , if fail, DECLINE immediately)

2. Credit Score Thresholds (SECOND , apply exact ranges, no judgment)

3. Affordability Checks (THIRD , DTI and disposable income)

4. Employment Stability (FOURTH , tenure requirements)

ABSOLUTE PROHIBITIONS:

, NEVER approve credit score 640, 699 (MUST be MANUAL REVIEW)

, NEVER approve credit score 620, 639 (MUST be MANUAL REVIEW)

, NEVER approve if ANY required document unverified

, NEVER use compensatory logic

, NEVER aggregate positive factors to override policy violation

CREDIT SCORE RULES (NO EXCEPTIONS):

, Below 640: DECLINED

, 640, 699: MANUAL REVIEW (not approved, even with perfect metrics)

, 620, 639: MANUAL REVIEW

, 700+: Can proceed to other checks

Result: 24 APPROVED, 39 DECLINED, 37 MANUAL REVIEW (78% agreement)

Improvement: +4% from iteration 1, +20% total from baseline

A.3 SESL Rule Generation Prompt

"You are an expert SESL rule, writer. Convert the policy below into SESL YAML only.

CRITICAL SESL SYNTAX RULES (follow exactly):

1. TOP, LEVEL STRUCTURE:

  , const: (optional) define reusable constants

  , rules: (required) list of rules

  , facts: (optional) test scenarios

2. RULE STRUCTURE , Each rule must have:

  , rule: unique_name

  , priority: number (higher fires first)

  , let: (optional) derived variables

  , if: list of conditions

  , then: list of actions (write to result.*)

  , because: "explanation text"

  , stop: true/false

3. LET BLOCK RULES (CRITICAL , prevents circular dependencies):

  , LET variables are evaluated in ORDER from top to bottom

  , A LET variable can ONLY reference:

   * Input facts (from applicant data)

   * Constants (from const section)

   * LET variables defined EARLIER in the SAME let block

  , NEVER reference a LET variable that is defined later

  , NEVER use function calls (no min, max, sum, len, etc.)

  , Use only arithmetic: +, , , *, /, parentheses

, Use boolean expressions for complex logic

4. AVOIDING CIRCULAR DEPENDENCIES:

, If you need min/max of multiple values, check each individually in IF conditions

, For spreads between values, calculate ALL differences explicitly

5. CONDITIONS (in IF blocks):

, Simple comparisons only: ==, !=, >, <, >=, <=, in, not in

, NO arithmetic in conditions

, Logic combinators: all/and, any/or

, Boolean LET variables MUST be compared explicitly

6. ACTIONS (in THEN blocks):

, ALL actions write to result.* namespace

, Use quoted strings, booleans (true/false), or numbers

7. FIELD NAMES:

, Use EXACT field names from the provided applicant data

, Do NOT invent field names or assume naming conventions

8. FORMATTING:

, Valid YAML only (2, space indentation)

, No markdown fences

, No comments in output

Generate SESL code for: {prompt}

A.4 Synthetic Data Generation Details

Input Files and Data Sources

Dataset:

- File: Loan_dataset.json
- Contents: 100 synthetic loan applications
- Fields per applicant (16 total): id, name,

A.4 Synthetic Data Generation Details

Input Files and Data Sources

Dataset:

- File: Loan_dataset.json
- Contents: 100 synthetic loan applications
- Fields per applicant (16 total): id, name, income, debts, credit_score, employment_years, rent, loan_amount, pay_slips_verified, bank_statements_verified, id_verified, dti_ratio, disposable_income, bureau_score_1, bureau_score_2, bureau_score_3
- Format: JSON array of objects

Results Dataset:

- File: Loan_dataset_Manual_Result.json
- Contents: 100 manually assessed decisions and reasons
- Format: JSON array of objects

Policy Document:

- File: Loan_Approval_SOP.pdf
- Size: 7,685 characters (after text extraction)
- Contents: Three, tier decision framework including debt, to, income ratio thresholds, minimum income and employment requirements, credit score cutoffs, and documentation/verification rules

SESL Rules:

- File: Task1_Loan_Approval_rules.sesl
- Contents: 15 SESL rules and 20 constants
- Format: YAML, based SESL model
- Development: Generated by LLM using standard prompt with policy document

Execution Scripts

Main Evaluation Script:

- File: Task1_Loan_Approval_simple.py
- Purpose: Executes both GPT, 4 and SESL evaluations for all 100 applicants

Output Files

GPT, 4 Results Cache:

- File: Task1_Loan_Approval_gpt.json
- Contents: 100 evaluations with applicant ID, name, decision, and reasoning
- Format: JSON array (~150 KB)

SESL Results Cache:

- File: Task1_Loan_Approval_sesl.json
- Contents: 100 deterministic evaluations with decision, explanation, and execution trace
- Format: JSON array (~80 KB)

Combined Results Files:

- File: All_Runs_Results.csv (GPT results, SESL results, manual results)
- File: Task1_Loan_Approval_results.json (fully merged results and metadata)

Acknowledgments and Disclosures

Conflict of Interest: The author is the creator of SESL and is the Founder of Oblongix Ltd, which owns the SESL intellectual property. SESL is released under an open, source non, commercial license.

Funding: This research was conducted as part of product research and development at Oblongix Ltd. No external funding was received.

Data Availability: Synthetic test scenarios used in evaluation are available at www.sesl.ai/paper/data or upon request for academic review purposes.

Code Availability: The SESL engine implementation is available under license by contacting SESL at https://www.sesl.ai for academic use and independent verification.