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) have capability for performing automated decision support tasks, but their stochastic outputs and vulnerability to hallucinations limit their suitability for high-stakes domains that require determinism and auditability. As an example, we present SESL, a deterministic rule-based expert system language designed for settings where explicit, interpretable decision logic is required. Rule-based systems such as SESL offers a human-readable Domain Specific Language, a forward-chaining engine with fixed execution semantics, and built-in explanation artefacts including rule-firing traces and dependency graphs. We propose a hybrid neuro-symbolic workflow in which LLMs assist with rule authoring while the Rule-based system performs all operational decision-making. Through experiments on synthetic loan approval, we show that a Rule-based system provides perfect determinism and eliminates errors and hallucinated explanations observed in LLM-only baselines. These results demonstrate that symbolic execution layers such as SESL may provide a practical pathway for trustworthy use of LLMs in regulated or safety-critical applications. Through out this paper we used the teams SESL and Rule-based system interchangeably.

---

1. Introduction

LLMs such as GPT-style models have demonstrated strong performance on knowledge-intensive tasks, natural language reasoning, and code generation. However, they remain stochastic generative models: the same prompt can yield different outputs, and these outputs may include confident yet fabricated facts or justifications. Studies show that chain-of-thought explanations generated by LLMs are unstable across samples and need not reflect the model's true reasoning [4–6,15].

Recent empirical studies have documented hallucination rates of approximately 1.75% in user-reported issues with AI-powered mobile applications, with significant business and legal consequences documented. These properties 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].

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. Businesses therefore require infrastructure that leverages the flexibility of LLMs while enforcing determinism and explainability.

Recent efforts to improve the governance of LLMs such as guardrails, safety layers, structured prompting, or post-hoc validation can reduce some 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, like SESL (Simple Expert System Language), are designed for this setting. It 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. Throughout this paper, we use SESL, a rule-based decision system, as a case study of 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.

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 and SESL executes decisions deterministically.

3. We evaluate the hybrid on synthetic data for loan, insurance and VAT tasks, showing perfect determinism and fidelity and elimination of hallucinated justifications.

---

2. Problem Definition and Motivation

Organizations deploying automated decision-making in high-stakes commercial domains such as lending, underwriting, fraud, 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 [4,5,15].

Traditional expert systems are deterministic but costly to author manually. Businesses need a hybrid architecture where decision logic is symbolic and deterministic, but models can be authored and maintained efficiently [11, 12-14,19]

---

3. Background and Related Work

3.1 LLM Risks in High-Stakes Domains

LLMs are known to hallucinate factual information [1–3]. Business hallucination benchmarks reveal significant variability across models and task types, with hallucination rates remaining a primary barrier to production deployment. Chain-of-thought reasoning is not necessarily grounded in internal model mechanics [4–6].

Multiple studies have demonstrated that chain-of-thought faithfulness varies substantially across tasks, with larger models sometimes producing less faithful reasoning. These risks make LLM-only systems unsuitable for high-stakes decisions.

3.2 Explainability and Governance

Regulators increasingly require transparency and traceability. Goodman and Flaxman's seminal 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. The European Parliament AI Act [32], OECD [7] and NIST [8] frameworks require trustworthy, governable AI. Rudin argues forcefully for inherently interpretable models for safety-critical tasks, noting that post-hoc explanations for black-box models may perpetuate bad 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. In contrast, LLMs store knowledge implicitly in neural weights and generate outputs through probabilistic pattern prediction, which makes them flexible but prone to variability and hallucination. While symbolic systems provide stable, auditable decision logic, 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 artefacts, and explicit design for LLM-assisted rule authoring and validation.

Symbolic expert systems offer inherent interpretability [19]. Neuro-symbolic systems combine neural and symbolic reasoning, leveraging the strengths of both [12–14]. Recent surveys of neuro-symbolic AI identify four main features: representation, learning, reasoning, and decision-making.

Research emphasizes that neuro-symbolic AI enhances interpretability, robustness, and trustworthiness while enabling learning from less data.

---

4. Hybrid LLM–SESL Method

4.1 Responsibilities

In the 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 artefacts and improving human understanding of rule-based decisions. Rule-based systems, 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 behaviour from influencing outcomes while still leveraging LLMs' strengths in model authoring and explanation.

LLMs can help:

- Generate initial rules
- Refactor and document rules
- Propose scenarios which can be tested with the rules
- Produce natural-language explanations of the execution of the rules

Deterministic Rule-based systems:

- Provide the structure and mechanisms for the rules and scenarios
- Executes deterministic decisions
- Validates rules
- Provides traceable, grounded explanations

In Rule-based systems LLMs do not produce final decisions. They operate as authors, editors, and communicators. Decision authority is intentionally delegated to the Rule-based system 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.
- Rule-based systems provide 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 comparison

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 therefore become financially burdensome. Moreover, the governance of such systems, still not fully understood, has already proven to be challenging, often requiring additional highly specialised technical expertise.

In contrast, rule-based systems are inherently low-cost. They operate with high speed, are straightforward to manage, and incur minimal ongoing 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. Although successful implementations exist, particularly in mission-critical contexts, the cost and complexity of this authoring process have inhibited broader, large-scale adoption [39].

The emergence of LLMs has radically 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.

---

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, 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 name** , a unique, human-readable identifier.
- **Priority** (optional) , resolves conflicts when multiple rules write to the same target.
- **IF condition** , a boolean predicate over facts, constants, or computed values.
- **THEN actions** , 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 using the SESL Language

```
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.

Execution Flow

The engine performs the following steps:

1. **Model loading and validation** - Rules, constants, and fact scenarios are parsed; invalid identifiers, malformed expressions, or missing paths are surfaced early.
2. **Iterative rule evaluation** - Each rule is evaluated in sequence. A rule either:
    - **matches**: condition evaluates to true → actions are applied,
    - **fails**: condition evaluates to false,
    - **errors**: invalid expression, missing operand, or unsafe reference.
3. **State updates and propagation** - When a rule fires, its actions modify the working fact state.
4. The engine repeats evaluation until:
    - no rule can change state further (fixed point),
    - a rule explicitly requires the model to stop iteration,
    - a safety iteration limit is reached.

Deterministic Conflict Handling

SESL supports configurable conflict resolution policies. All policies are deterministic, meaning identical inputs always produce identical outputs.

Trace and Dependency Recording

Each rule evaluation is logged for debugging and explanation.

Strict Mode Safety Features

SESL's strict execution modes ensure reliability :

- unknown identifiers produce immediate errors,
- unquoted text cannot silently become a string,
- unsafe constructs (function calls, external references) are rejected,
- LET expressions are evaluated through a restricted AST interpreter preventing arbitrary code execution.

This guarantees that SESL models cannot behave unpredictably, even if authored or modified by LLMs.

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 capturing:

Rule Firing Traces - A chronologically ordered list of all rules evaluated, highlighting which rules fired, which failed and why, which conditions were met or unmet.

Driver Trees - A causal graph mapping how input facts led to final outputs, showing which rules contributed to each result value, and dependencies between facts, conditions, and outcomes. Driver trees are especially valuable for regulatory inspection, internal audit, and user-facing explanations.

Explanation Blocks - Human-readable summaries of matched conditions, evaluated LET expressions, assigned result values, reasons derived from rule definitions.

Execution Metrics – Includes number of iterations, rules matched/fired, performance metrics, conflict resolution events.

Together, these artefacts enable step-by-step reconstruction of the decision process, meeting governance obligations for transparency.

5.5 Tooling and Expert Validation Workflow

The SESL CLI (command-line interface) provides a professional-grade environment for model development and governance. The tooling includes:

- **Interactive execution mode** - Enables step-by-step evaluation of rules, useful for debugging and training.
- **Batch execution mode** - Supports CI/CD pipelines, regression testing, and bulk scenario runs.
- **Linting and structural validation** - Detects: unreachable rules, unused constants, conflicting assignments, missing fact paths, unintended model behaviours.
- **Scenario testing framework** - Allows large suites of test inputs to validate policy behaviour across edge cases.
- **Dependency graph generation** - Produces visualizations showing: rule dependencies, fact-to-result flows, circular or redundant logic.

5.6 Expert Review and Governance

SESL fits naturally into business governance workflows:

- Business experts review rule text because it is human-readable.
- Engineers validate execution traces for technical correctness.
- Risk and compliance teams audit models using SESL's deterministic logs.

SESL includes inbuilt descriptive metadata (e.g. data sources, creation/update dates, owner, etc.) including version control ensuring traceability of every rule modification.

LLMs may assist in drafting or refactoring rules, but SESL remains the authoritative execution environment. This makes SESL suitable for large, regulated organizations that require demonstrably transparent and reproducible decision-making systems.

---

6. Testing 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 receives both the natural-language policy description and a structured case profile. The LLM is asked to determine the correct decision and to 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 is called using the details for each applicant as a scenario, it then evaluates each case deterministically using its forward-chaining rule engine. Explanations are produced procedurally from SESL's rule-firing trace.

6.2 Experimental Tasks and Evaluation Procedure

We evaluated system performance across three representative high-stakes decision processes: loan eligibility assessment, insurance risk tier assignment, and VAT rate selection. Each reflects a well-structured business workflow where decisions are governed by policy, thresholds, exceptions, and regulatory constraints.

Task 1 , Loan Eligibility Assessment

This task models a lending policy that incorporates income thresholds, debt-to-income ratios, credit scores, and employment stability. Each scenario represents a synthetic and simplified loan application to be evaluated.

Typical Business Process (Based on common regulatory lending workflows, e.g., UK FCA Responsible Lending Guidelines):

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.

This process aligns with industry-standard lending workflows emphasising transparency and responsible decision-making [20].

Method

In this task we take a file of applicant information, and a policy document as input (note the manual determination of the result is not included in the data at this point). For SESL we create the rules from the policy document, and then for each applicant the scenario using the LLM from the applicant data. For each of the 100 applicants we then (1) Using a simple prompt ask GPT4o for the decision and reason, and (2) Ask SESL to produce the decision and reason. We repeat this process 5 times saving the output from both in separate JSON files per run.

A final program merges the separate JSON files, with the orgianal data, and the manually assessed results to give a single CSV file for review and analysis.

A full list of the prompts and programs used can be found in Appendix 1.

6.3 Data

For each of the three decision-making tasks we generated 100 synthetic test cases (Scenarios) using realistic but non-sensitive distributions for all input features. These datasets ensure coverage of typical scenarios as well as edge cases, enabling consistent evaluation across all system configurations.

The data was then annotated manually with the expected outcomes needed for the testing. All of the data (and testing programs and results) can be found on the www.sesl.ai website. A full list of the data is in Appendix 1.

6.4 Experimental Results Summary

We evaluated both approaches across 600 synthetic scenarios (100 per task), conducting six repeated runs per scenario to assess determinism and stability. After the first and fourth runs, we applied prompt-tuning to the GPT-based system to improve performance; the tuning procedures are described in Appendix 1.

Table 1 reports the distribution of decisions produced by GPT, SESL, and the manually validated benchmark.

Table 1 – Distribution of Decisions

Decision	Count of Decision			% of Total		
	GPT	SESL	Manual	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%
Grand Total	600	600	600	100%	100%	100%

Across all test conditions, SESL more closely 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; notably, the use of the LLM to assist in the drafting of SESL rules helped overcome a historically significant barrier associated with manual rule authoring. While good news it was disappointing not to measure some volatility, however we decided to proceed with the experiment.

GPT, by contrast, deviated from the manual benchmark in 78 cases when aggregated across the full set of scenarios. While GPT's distributional mismatch does not equate to item-level accuracy, it does indicate that its outputs diverge materially from the expected manual decision pattern. SESL, on the other hand, matched the manual distribution exactly for all three decision categories.

We tested whether GPT decision distribution (Approved, Declined, Manual Review) differs significantly from the manual distribution, with a Chi-square statistic: 14.36 and p-value: 0.00076, the p-value is well below 0.05, meaning GPT's distribution is statistically significantly different from the manual benchmark.

Table 2 – Distribution of Decisions by Run to determine accuracy

Decision	GPT Count of 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
Grand Total	100	100	100	100	100	100	600

Decision	SESL Count of 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
Grand Total	100	100	100	100	100	100	600

Table 2 presents the distribution of decisions across six repeated GPT runs and six SESL runs, each applied to identical scenario inputs. As expected for a deterministic rule-based system, SESL produced identical outputs across all runs, yielding perfect consistency for each decision category (Approved, Declined, Manual Review).

By contrast, the GPT-based system exhibited run-to-run variability, even though the input data and original prompt remained unchanged (in runs 1,2,3 and 4). This variation is characteristic of stochastic large language models, which do not guarantee deterministic outputs without explicit constraints.

Notably, the differences across the six GPT runs were not statistically significant (Chi-square statistic: 13.66, $p = 0.189$), indicating that the observed fluctuations fall within the range expected by chance when sampling from a probabilistic model. However, the magnitude and direction of changes between specific runs correspond to the points at which prompt adjustments were introduced (between runs 0 and 1, and again between runs 4 and 5). These tuning steps produced substantial improvements, bringing GPT's decision distribution into closer alignment with the manually validated benchmark after the first tuning round.

Together, these results illustrate two important dynamics:

1. Determinism vs. Variability - SESL reliably reproduces the manual process across all repetitions, whereas GPT—even with identical inputs—exhibits inherent variability that must be managed.
2. Sensitivity to Prompt Design - Modifying the GPT prompt had a measurable and meaningful impact on performance, demonstrating the critical role of prompt

engineering in achieving acceptable levels of accuracy and distributional alignment.

Table 3 – GPT, SESL, Manual consistency

Applicant Id	Decision	run_0	run_1	run_2	run_3	run_4	run_5
8	MANUAL REVIEW				1	1	1
	APPROVED	1	1	1			
16	MANUAL REVIEW	1	1				1
	DECLINED			1	1	1	
18	MANUAL REVIEW			1	1	1	1
	APPROVED	1	1				
51	APPROVED	1		1			
	MANUAL REVIEW		1		1	1	1
55	DECLINED				1		1
	APPROVED	1	1	1		1	
64	MANUAL REVIEW	1	1				
	DECLINED			1	1	1	1
66	MANUAL REVIEW		1				
	DECLINED	1		1	1	1	1
94	MANUAL REVIEW	1			1		1
	DECLINED		1	1		1	

To assess GPT's decision stability (volatility), we analysed how often an applicant received *the same* outcome across runs 1 - 4, during which neither the prompt nor the input data were modified. Under such controlled conditions, a consistent model should return identical classifications for each applicant across all four runs.

As shown in Table 3, eight applicants (out of 100) exhibited at least one change in decision across these runs, with the number of inconsistencies distributed as follows: five changes in run 1, four in run 2, two in run 3, and three in run 4 (each run being 100 independent tests). This give an approximate volatility of 3-5%.

Although the overall level of variation is small relative to the total number of classifications, these fluctuations demonstrate that GPT introduces a measurable degree of non-determinism even when all experimental conditions remain fixed.

7. Discussion

Our evaluation highlights complementary strengths across symbolic and LLM approaches and shows 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 behaviour 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. Taken together, the two approaches form a hybrid architecture that balances flexibility with reliability.

The key insights emerging from this comparative analysis are summarised below.

7.1 Key Insights

1. **Determinism and fidelity require symbolic execution.** Our results show that a symbolic rule-based engine can guarantee that the same inputs always lead to the same outputs. Deterministic evaluation is essential for business governance, auditability, and policy fidelity - requirements that LLMs alone cannot consistently satisfy.
2. **Explanations are most reliable when derived from symbolic traces.** Explanation artefacts like those generated by SESL (rule traces, driver trees, monitor blocks) are faithful, complete, and reproducible. In contrast, LLM-generated natural-language justifications frequently omit details or hallucinate unsupported reasoning steps, consistent with research showing that chain-of-thought explanations may not reflect actual model reasoning processes.
3. **Hybrid architectures achieve both flexibility and reliability.** LLMs are powerful policy interpreters and model authors, but rule-based systems provide the deterministic substrate that executes decisions safely. Combining the two yields a system that is expressive yet governed, adaptive yet controlled.
4. **LLMs alone are highly effective model authors but poor decision engines.** LLMs excel at transforming policy text into structured rules but should not be entrusted with final decision-making. Their stochasticity and reasoning instability make them well-suited for authoring logic, documentation, and scenarios, but not executing policy.
5. **Deterministic rule engines form a natural compliance and audit surface.** Rule-based systems explicit rules, conflict policies, and traceability align with regulatory frameworks [7,8]. Symbolic reasoning creates artefacts that auditors can inspect and validate, bridging the gap between AI and governance

requirements. This addresses regulatory requirements for meaningful information about algorithmic decision-making processes.

6. **Scenario-based testing becomes a powerful governance mechanism.** Because a Rule-based system is deterministic, scenario libraries behave as executable specifications. They enable regression testing, policy drift detection, version control of rules, and automated compliance assurance. LLMs further enhance this workflow by proposing new scenarios and edge cases.
7. **Hybrid neuro-symbolic designs reduce hallucinations without reducing expressiveness.** Routing policy execution through a symbolic engine forces LLM outputs into structured, verifiable rules. This bounded symbolic channel dramatically reduces hallucination and ensures that explanations and decisions remain tied to explicit logic.
8. **Symbolic substrates enhance long-term maintainability and organizational memory.** Rules explicitly encode the rationale behind decisions, making models easier to understand, audit, modify, and transfer across teams. Unlike black-box neural systems, symbolic rule sets persist as stable institutional assets.

7.2 Limitations

Real-world rule based system models may scale to thousands of rules

In large business, operational decision engines often encode decades of policy evolution, regulatory constraints, product variations, and exception handling. A mature deployment of a rule based system may therefore require managing thousands of rules across multiple interacting models. While having a structured language, linter, and scenario-based testing are designed to support this scale, the cognitive and organizational complexity of maintaining such 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 Rule-based decision tools in business environments.

LLM prompting requires governance

Although LLMs can dramatically 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.

As a result, LLM use in rule based system workflows must operate within a formal governance framework that includes versioned prompts, validation checkpoints, reproducibility controls, and mandatory human review. Organisations should treat LLM-generated rules as suggestions rather than executable policy until they have passed automated linting, regression testing, and expert verification. Governance around LLM prompting is therefore essential to prevent unintentional policy drift and ensure regulatory compliance.

Human review remains essential

Despite the deterministic semantics and comprehensive explanation capabilities, the correctness of a rule based 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. Rule-Based systems therefore enhance, but does not replace, the human governance processes required for safe and responsible AI deployment in high-stakes domains.

Business integration complexity

Deploying rule based system 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 the rule based system, the LLM services, and operational databases can be non-trivial, particularly in environments with strict security and governance constraints. Successful business adoption therefore depends on careful architectural planning, robust DevOps practices, and comprehensive testing of end-to-end decision flows.

The Rule Authoring Bottleneck

While rule-based systems 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

labelled data alone. Organisations 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 of workflows.

Comparison with Modern Interpretable ML

Our evaluation does not include comparisons with other ML based 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. Rule-based systems are 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 a Rule-based system 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, Rule-based systems reduce 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. A Rule-based system 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, having a Rule-based system 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 into such a system as rules, and these may reflect historical inequities or incorrect assumptions.

While a Rule-based system 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 a Rule-based systems 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. Overall, the SESL + LLM hybrid approach provides a strong foundation for safer business AI, but it must be embedded within a comprehensive governance strategy to realize its full societal benefits.

7.4 Decision Framework: When to Use a Rule-based System vs. Alternatives

A Rule-based system is not universally optimal. We recommend 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:

LLM's and Rule-based Systems will not solve the broad range of challenges alone. When considering how to approach a solution there are more methods and approaches which should be considered :

- 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

Potential Positive Impacts:

- Increased transparency in automated decision-making
- Reduced risk of opaque algorithmic harm
- Enhanced regulatory compliance and public trust
- Preservation of institutional knowledge in interpretable form

Potential Negative Impacts:

- False sense of security: Interpretable ≠ fair or correct
- Regulatory arbitrage: Organizations might use SESL for compliance theater while actual decisions remain opaque
- Automation of discrimination: Rules can encode bias as easily as ML models
- Reduced innovation: Explicit rule systems may discourage exploration of novel decision patterns

Deployment Recommendations:

1. SESL should complement, not replace, fairness auditing
2. Organizations must test for disparate impact regardless of interpretability
3. Expert review should include diverse perspectives, not just technical validation
4. Continuous monitoring for unintended consequences is essential

Dual-Use Concerns:

While SESL is designed for legitimate business governance, it could potentially be misused for:

- Automating discriminatory decisions with plausible deniability
- Encoding ethically questionable policies in "interpretable" form
- Circumventing algorithmic accountability requirements through technical compliance

We advocate for SESL deployment only within comprehensive AI governance frameworks that prioritize fairness, accountability, and 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 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.

Rule-based Systems such as SESL are not universal solutions. It is most appropriate for process and regulated domains where explicit policy encoding, direct expert control, and pre-hoc interpretability are required or strongly preferred. Organisations should evaluate Rule-based approaches 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 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 Rule-based Systems with learned components

We believe Rule-based systems such as 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 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, fidelity, hallucination detection) 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 supplementary materials to enable replication.

Reproducibility Note: LLM outputs may vary across API versions even with fixed parameters. Our evaluation used chatGPT4o. 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*.

[24] SESL Engine Source Code. (2025). rule_engine.py. SESL documentation at https://www.sesl.ai

[25] SESL CLI User Guide. (2025). https://www.sesl.ai

[26] SESL Language/User Guide. (2025). https://www.sesl.ai

[27] SESL Product Summary Guide. (2025). https://www.sesl.ai

[28] ISO. (2018). *ISO 31000:2018 Risk Management , Guidelines*. International Organization for Standardization.

[29] European Parliament and Council. (2016). Regulation (EU) 2016/679 (General Data Protection Regulation). *Official Journal of the European Union*, L119, 1-88.

[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)

[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 for GPT reponse

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 (processed credit checks despite unverified docs)
- Aggregated multiple positive factors to override single policy violation

Prompt Engineering Iteration 1 (run_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 (from run_1-4 disagreement analysis):

- Still auto-approving 6 manual review cases (credit score 640-699)
- Too lenient on 10 document verification failures (saying "strong profile" overrides missing docs)
- Misinterpreting credit 620-639 range (should be MANUAL REVIEW, GPT sometimes approves)
- No explicit rule execution order (checks credit before documents)

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 (high income [1] override credit)

- 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 (74% ® 78%), +20% total from baseline (58% ® 78%)

A.3 SESL rule generation prompts and SESL

"""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: and, or, == (do NOT use standalone 'not')

  EXAMPLE - CORRECT ordering:
  let:
    base_amount: loan_amount * 0.8   # Uses input fact
    monthly_payment: base_amount / 12   # Uses earlier LET variable
    has_income: monthly_income > 3000   # Boolean expression (OK in LET)
    strong_profile: credit_score >= 720 and income >= 50000 # Complex boolean (OK in LET)

  EXAMPLE - WRONG (circular dependency):
  let:
    spread: max_score - min_score # BAD: references min_score before it's defined
    min_score: bureau_score_1   # BAD: defined after being used

  EXAMPLE - WRONG (function calls):
  let:
    min_score: min([score1, score2]) # BAD: no function calls allowed
    max_spread: max(spreads)   # BAD: no function calls allowed

4. AVOIDING CIRCULAR DEPENDENCIES:
  - If you need min/max of multiple values, check each individually in IF conditions
  - Instead of: let: min_val: min([a,b,c])
    Use in IF: if: any: [a < threshold, b < threshold, c < threshold]
  - For spreads between values, calculate ALL differences explicitly:
    spread_1_2: val1 - val2
    spread_2_1: val2 - val1 (for absolute difference)
  - For NOT logic, create a positive LET variable instead:
    Instead of: if: not (score >= 720 and income >= 50000)
    Use: let: strong_profile: credit_score >= 720 and income >= 50000
        if: strong_profile == false

5. CONDITIONS (in IF blocks):
  - Simple comparisons only: ==, !=, >, <, >=, <=, in, not in
  - NO arithmetic in conditions
  - NO standalone boolean variables (always use explicit comparison)
  - Logic combinators: all/and, any/or (do NOT use 'not' operator)
  - Reference: input facts, constants, or LET variables
  - CRITICAL: Boolean LET variables MUST be compared explicitly
    WRONG: if: bureau_1_low
    RIGHT: if: bureau_1_low == true
    BETTER: Skip LET, use direct comparison in IF

  EXAMPLE - CORRECT conditions:
  if:
    all:
      - credit_score > 640   # Direct comparison (BEST)
      - income >= 30000   # Direct comparison (BEST)
      - strong_profile == true   # Boolean LET variable with explicit comparison

  EXAMPLE - WRONG conditions:
  if:
    all:
      - bureau_1_low   # BAD: No explicit comparison
      - not (score >= 720)   # BAD: 'not' operator not supported
      - income + bonus > 50000   # BAD: Arithmetic in condition

6. ACTIONS (in THEN blocks):
  - ALL actions write to result.* namespace
  - Example: result.decision: "APPROVED"
  - Example: result.reason: "Credit score meets threshold"
  - 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
  - Common mistakes to avoid:
    * Don't use 'credit_bureau_1' if data has 'bureau_score_1'
    * Don't use 'verified_*' if data has '*_verified'
    * ALWAYS check the provided field list carefully

8. FORMATTING:

- Valid YAML only (2-space indentation)
- No markdown fences (```yaml or ```)
- No comments in output
- Lists use dashes (-)

Generate SESL code for: {prompt}
"""

Appendix A.4 — Synthetic Data Generation Details

A.4.1 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 synthetic loan application decision and reasons
- Manually authored
- Format: JSON array of objects

Policy Document

- File: Loan_Approval_SOP.pdf
- Size: 7,685 characters (after text extraction)
- Contents: Three-tier decision framework (APPROVED / DECLINED / MANUAL REVIEW) including:
  - Debt-to-income ratio thresholds
  - Minimum income and employment requirements
  - Credit score cutoffs
- Documentation and verification rules

SESL Rules

- File: Task1_Load_Approval_rules.sesl
- Contents: 15 SESL rules and 20 constants
- Format: YAML-based SESL model (const, rules sections)

- Development Notes:
    - Generated by LLM using standard prompt with policy document
    - No circular dependencies
    - All field names verified against dataset schema

A.4.2 Execution Scripts

Main Evaluation Script

- File: Task1_Load_Approval_simple.py
- Purpose: Executes both GPT-4 and SESL evaluations for all 100 applicants

---

A.4.3 Output Files

GPT-4 Results Cache

- File: Task1_Load_Approval_gpt.json
- Contents:
  100 evaluations including:
    - Applicant ID
    - Name
    - GPT-generated decision
    - Natural-language reasoning
- Format: JSON array
- Size: ~150 KB (varies with reason length)

SESL Results Cache

- File: Task1_Load_Approval_sesl.json
- Contents:
  100 deterministic SESL evaluations including:
    - Decision
    - Explanation
    - Full execution trace (facts_after)
- Format: JSON array
- Size: ~80 KB

Combined Output and Manual results CSV

- File: All_Runs_Results.csv
- Contents: Fully merged results and metadata
- Fields:
    - gpt_results

    - sesl_results
    - Manual results

Combined Results File

- File: Task1_Load_Approval_results.json
- Contents: Fully merged results and metadata
- Fields:
    - gpt_results
    - sesl_results
    - policy_source
    - applicants_count

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.