Deterministic Compliance Failures in Large Language Models: A Structural Analysis Using a Legacy Fuzzy Inference Benchmark

Toyoji Kanagawa
Independent Researcher, Japan

engrXiv Preprint, March 2026

Abstract

About forty years ago, the author implemented a fuzzy inference engine in C on an MS-DOS platform as part of a master's thesis on PID auto-tuning control. In 2026, an attempt to reconstruct that system through LLM-assisted pair programming failed: despite detailed specifications and confirmed comprehension, the model repeatedly and autonomously rewrote deterministic logic, rendering the codebase unrecoverable. The author completed the implementation alone. This experience generated a hypothesis — that large language models lack the architectural capacity for deterministic specification compliance — and motivated the formal investigation reported here.

Six state-of-the-art LLMs were evaluated on their ability to execute the original fuzzy inference specification through zero-shot reasoning, without code generation. The task required faithful table lookup, discrete state maintenance, priority-ordered early-exit logic, and mandatory multi-label aggregation — properties that are, in several critical respects, antithetical to autoregressive language generation. A structured white-box output format was mandated, enabling precise localization of any deviation to a specific inference step.

A follow-up replication study confirmed these findings, revealing that even when a model acknowledges its prior errors or demonstrates meta-cognitive understanding of the rules, it consistently fails to maintain deterministic execution. One model's failure mode even evolved from passive evidence suppression to active data tampering—arbitrarily re-

This manuscript is an updated version of a preliminary record previously deposited on Zenodo. The current version reflects significant revisions, including the results of a second follow-up test and refined structural analysis.

interpreting input values to justify a biased output. Five of six models failed to achieve full compliance. The failures were not random: seven categorically distinct failure mechanisms were identified across nine observed failures, including ancillary condition misread, multi-label detection failure, table misread, probabilistic override of explicitly dominant membership grades, aggregation rule bypass, output element hallucination, and termination rule violation driven by format-compliance rationalization. Critically, the model that recorded a 0% pass rate in the formal evaluation was the same model whose pair-programming behavior had originally motivated this research — a convergence of experimental data and practical observation that the author terms longitudinal validation. One model, Claude Sonnet 4.6, achieved perfect compliance across all test cases, exhibiting none of the identified failure modes. Its behavioral profile suggests a qualitatively different relationship to specification documents — one in which explicit instructions function as binding constraints rather than contextual suggestions.

The findings support a definitive engineering conclusion: LLMs that exhibit any of the identified failure modes under controlled evaluation conditions are categorically unsuitable for autonomous deterministic automation, regardless of general capability ratings. The boundary of responsible LLM deployment is not defined by average performance. It is defined by the nature of the failures that occur at the margin.

1. Introduction

1.1 Background

Large language models (LLM) have been rapidly integrated into software engineering workflows, including automated code generation, specification interpretation, and AI-assisted development. This integration has prompted growing interest in the boundaries of LLM reliability — particularly in domains where output correctness is not gradable but binary. Control engineering and other mission-critical fields represent precisely such a domain: deterministic logic must be executed with zero tolerance for deviation, and a plausible but incorrect output is indistinguishable from a dangerous one.

1.2 Research Objective

This study evaluates whether state-of-the-art LLMs can execute a fully specified, unambiguous deterministic algorithm with complete procedural fidelity — not as a code generation task, but as a direct reasoning task in which the model itself serves as the execution engine.

1.3 Problem Statement

LLMs are optimized to generate contextually plausible outputs, a property that is in fundamental tension with deterministic specification compliance. Even when presented with explicit lookup tables, discrete membership grades, and unambiguous branching conditions, an autoregressive model has no architectural guarantee that retrieved values will be preserved and propagated correctly rather than overwritten by distributional pressure. This study examines whether that tension produces observable, classifiable failures — and whether those failures are severe enough to disqualify LLMs from deterministic automation roles.

1.4 Motivation: Lessons from a Personal Reproduction Project

This research originated from a personal project to reproduce an automated PID parameter tuning system using fuzzy inference, originally developed by the author about 40 years ago using C on MS-DOS [7]. The original implementation, documented in Kanagawa (1987), provided a fully specified deterministic rule base for fuzzy-inference-driven PID gain adjustment.

During pair-programming sessions with Generative AI (mainly Gemini Flash), a critical issue emerged. Despite providing detailed specifications and confirming the AI's understanding, the model repeatedly and autonomously altered the code logic against the deterministic decision trees defined in the original work. This persistent unauthorized rewriting forced the author to abandon AI-assisted coding and complete the implementation manually, producing a modern React/JavaScript reimplementation of the 1987 controller.

This experience raised a fundamental hypothesis: Generative AI lacks the discipline required for deterministic reasoning based on label interpretation, such as fuzzy inference. This study was designed to verify this hypothesis.

2. Related Work

This study draws on three bodies of literature, each of which it extends in a specific direction.

Fuzzy inference and control. The theoretical foundation of this work rests on Zadeh's (1973) formalization of fuzzy set theory and its subsequent application to industrial control by Mamdani and Assilian (1975). The initial PID parameters in the evaluated system are established using the discrete-time tuning method of Takahashi and Chan (1971), which provides a theoretically sound baseline from which fuzzy-based corrections are applied. The inference engine itself is based on a fuzzy PID auto-tuning system originally implemented by the author in 1987 and documented in Kanagawa (1987). Its role in this study is not to advance control theory, but to serve as a fully specified, ground-truth-verified computational pipeline against which LLM compliance can be measured with sub-step precision.

LLM reasoning limitations. Wei et al. (2022) demonstrated that chain-of-thought prompting improves multi-step reasoning, but also revealed that systematic reasoning in LLMs is elicited rather than intrinsic. The Apple (2025) study found that apparent reasoning capability does not reliably generalize as structural complexity increases — a finding this study corroborates and extends by demonstrating that failure modes reflect categorically distinct mechanisms of specification violation, not merely quantitative accuracy degradation.

Instruction following and compliance. Prior instruction-following evaluations, including Zeng et al. (2023) [6], have applied gradable correctness criteria, where partial compliance yields partial credit. This study applies an engineering-grade zero-tolerance criterion — standard in safety-critical systems — which this literature has not previously examined. The two-part prompt structure employed here, separating specification comprehension from execution, enables precise localization of failure to one of these two distinct stages.

To the author's knowledge, no prior work has evaluated LLM compliance using a legacy industrial algorithm as the benchmark, with full ground-truth verifiability at every intermediate step. This configuration is the methodological contribution of the present study.

3. Experimental Methodology

3.1 Engineering Foundation: I-PD Control System

To establish a clear ground truth for the evaluation, this study utilizes the industrial-grade digital control system shown in Figure 1. This system represents the original application for which the author's fuzzy inference engine was developed (1987).

3.1.1 Control System Configuration

The simulation employs a Digital I-PD Control structure. As illustrated in the block diagram:

- Reference Input (0): Maintained at zero to focus on the disturbance response.
- Integral Controller ($Ki\frac{Z}{Z-1}$): Processes the error E(z) and drives the steady-state error toward zero.
- IPD Structure: Unlike standard PID, the Proportional (Kp) and Derivative (Kd (1-z^{-1})) terms are placed in the feedback path. This configuration is specifically designed to suppress proportional kick, ensuring smoother response during parameter adjustments.
- Disturbance (V(z)): A step-like disturbance is injected to test the system's resilience.

In this study, the initial values of Kp,Ki and Kd are calculated and set using the Takahashi-Chan tuning method [9]. This ensures that the evaluation begins from a theoretically sound baseline, where any subsequent fuzzy-based corrections (ΔK) are intended to further optimize an already standard-compliant loop.

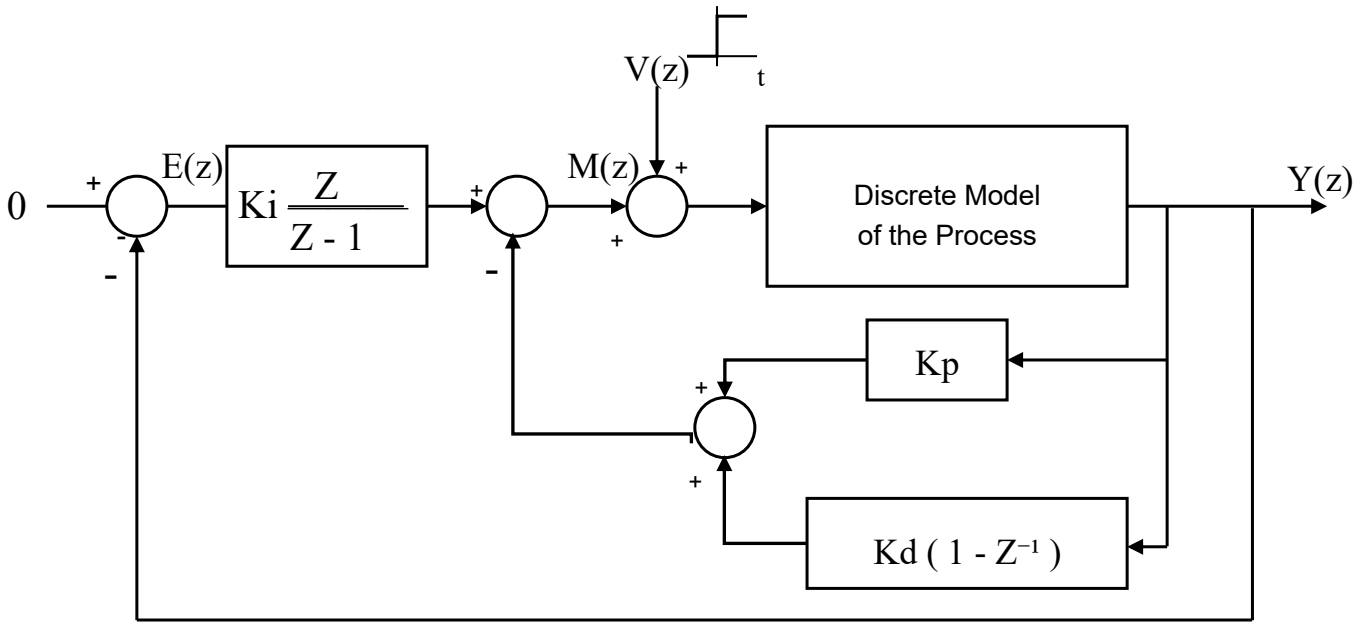

Fig. 1 Control System Configuration used in Simulation

3.1.2 Defining the AI's Role: Logic Execution, Not Dynamic Control

It is crucial to define the scope of the AI's task within this architecture. The LLM is not functioning as the controller itself, nor is it performing real-time optimization of the process.

Instead, the AI is tasked with executing the Fuzzy Inference Logic that determines the parameter corrections (ΔKp, ΔKi, ΔKd) based on observed waveform characteristics. In the original system, this logic is a hard-coded, deterministic algorithm. By asking the AI to replicate this inference, we are testing whether the LLM can maintain 100% procedural fidelity to a specification that is essential for the stability of the physical system depicted in Figure 1.

3.1.3 Optimization Metric: Area Decay Ratio

The success of the control is measured by the Area Decay Ratio, calculated from the response curve. The inputs provided to the AI are five specific ratios, α1 through α5, which represent the areas under the deviation curve.

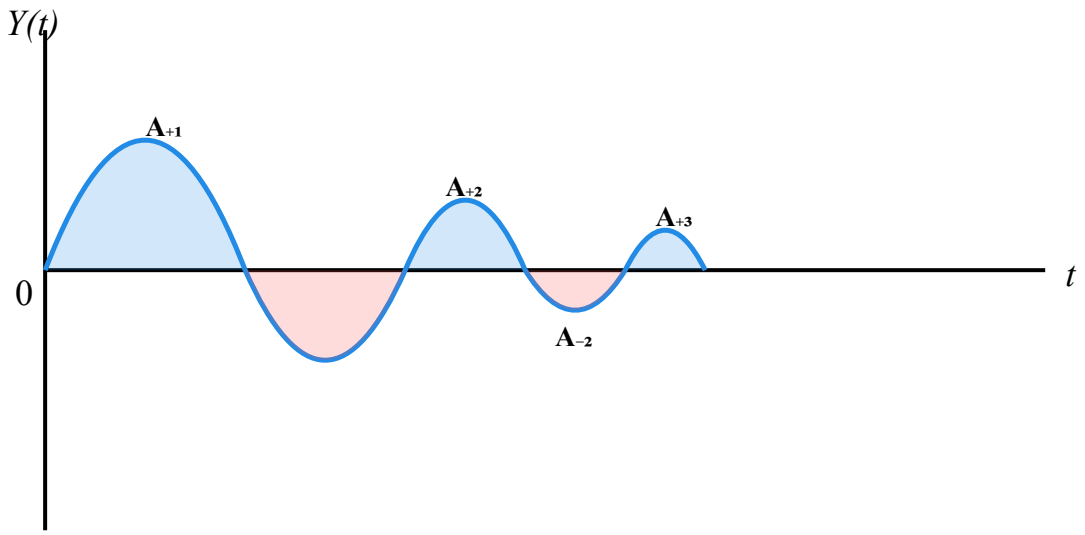

Fig. 2 Definition of Deviation Areas

$\alpha_1 = (|A_{-1}| / |A_{+1}|) \times 100$ [%]
$\alpha_2 = (|A_{+2}| / |A_{+1}|) \times 100$ [%]
$\alpha_3 = (|A_{-2}| / |A_{-1}|) \times 100$ [%]
$\alpha_4 = (|A_{+2}| / |A_{-1}|) \times 100$ [%]
$\alpha_5 = (|A_{+3}| / |A_{+1}|) \times 100$ [%]

3.2 Reference Implementation (Ground Truth)

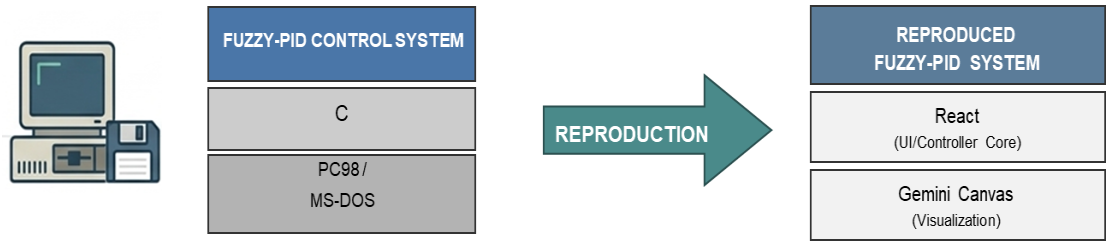

Fig. 3 Legacy Reproduction

This experiment does not seek an approximate or suggested solution. The author has already reconstructed the 1987 legacy MS-DOS/C implementation documented in Kanagawa (1987) into a modern React-based controller. This reference system provides the absolute Ground Truth for every test case. The AI is expected to reproduce these deterministic results through zero-shot specification execution.

3.3 Fuzzy Inference Engine Specification

The AI is provided with a complete Fuzzy Inference specification, which must be executed as a strictly deterministic computational pipeline. This process is not a creative reasoning task but a sequence of specified operations where any deviation at any stage is defined as a failure. As illustrated in the Fuzzy Inference Logic Flowchart (Fig. 4), the pipeline consists of four distinct stages:

1. Fuzzification and Element Indexing:

The five area damping ratios (α1,…,α5) are mapped to fuzzy subsets by identifying the Element Index (the specific row number in the Universe of Discourse table). The model must accurately retrieve the membership grade (μ) from this index. Any error in index selection at this stage propagates as a structural failure through the entire pipeline.

2. Sequential Inference (Stages I→ II →III or IV):

The model must navigate the priority-ordered logic flow. This includes monitoring Early Exit conditions (e.g., μ>= 0.8 in Stages I, II) and executing conditional branching based on the relation α2>=α1.

3. Aggregation (Multi-Rule Merging):

In cases where multiple rules are fired, the model is required to perform a mandatory arithmetic mean of the corresponding correction values.

4. Defuzzification:

The final fuzzy conclusion is converted into a unique, crisp correction value (e.g., ΔKp). This numerical output is the final command for the I-PD control loop.

Detailed definitions of the membership functions and the complete rule base are provided in Appendix B.

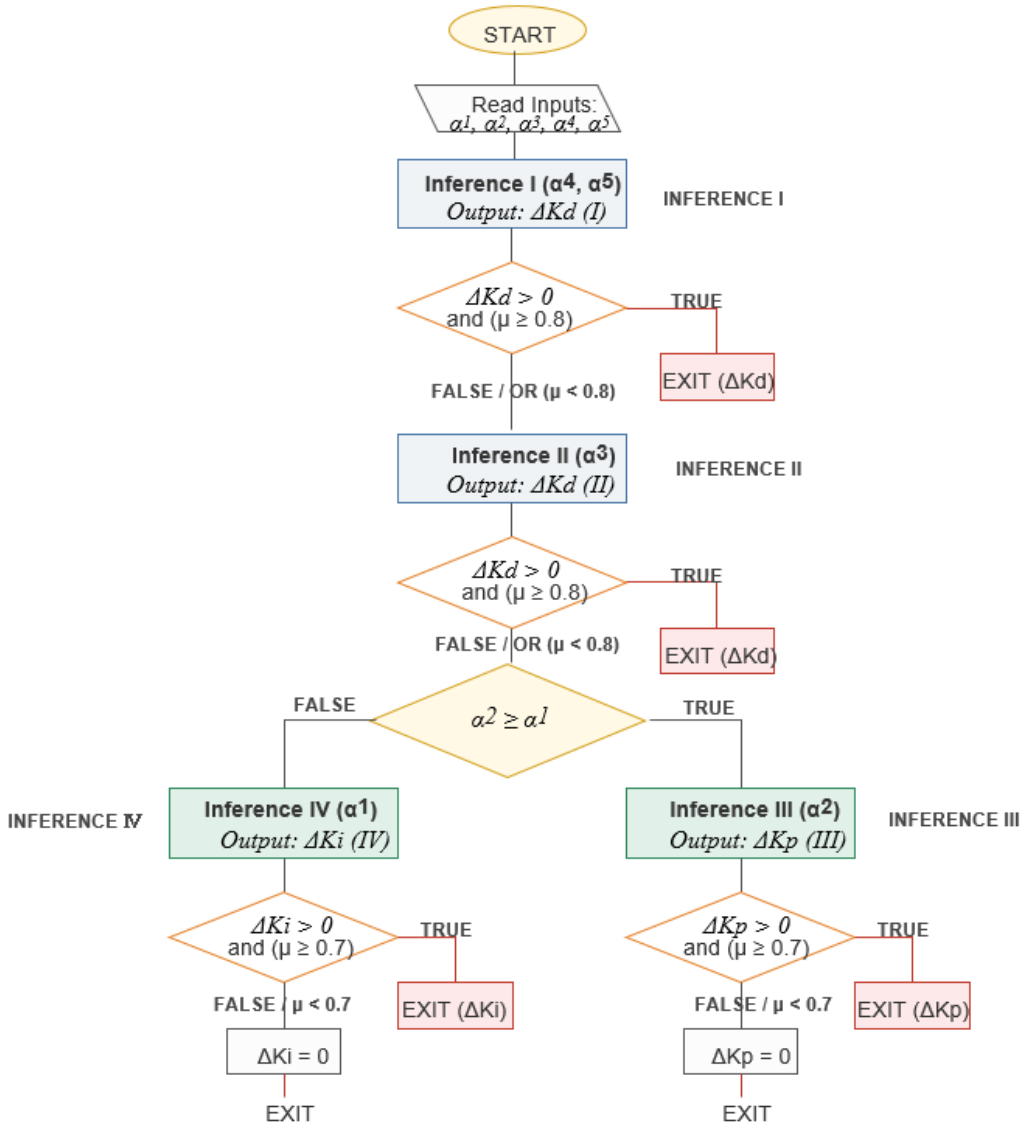

Fig. 4 Fuzzy Inference Logic Flowchart

3.4 Why Fuzzy Inference Constitutes an Ideal Stress Test

Fuzzy inference systems (FIS) represent a particularly demanding class of computational tasks for large language models (LLMs), not in spite of their apparent simplicity, but by virtue of their structural properties. Unlike open-ended reasoning tasks — where partial correctness or plausible approximation may be acceptable — a FIS requires the executing agent to follow a strictly ordered, fully deterministic procedure in which every intermediate step is verifiable and every deviation is unambiguous.

The inference engine evaluated in this study was originally implemented in C on an MS-DOS platform in 1987 and reconstructed in React for the present investigation. It accepts five physical input variables (α_1–α_5), derived as area decay ratios from a closed-loop step response, and produces a single corrective output — ΔKp, ΔKi, or ΔKd — through a priority-ordered, multi-stage inference pipeline. The pipeline terminates immediately upon the first confident output, and the correct termination point differs across cases. This architecture ensures that no two test cases exercise the same inference path, providing structural coverage of the decision space without requiring a large number of test instances.

3.5 Test Case Design and Structural Coverage

This study employs a Structural Coverage approach in place of large-scale statistical sampling. Four test cases were engineered to activate every unique decision path in the inference pipeline, providing complete coverage of the decision space with a minimal yet exhaustive test suite.

Each case targets a distinct inference trajectory:

- **Test 1 (I → II → IV):** Long-range sequential logic and conditional branching ($\alpha 2 < \alpha 1$) leading to ΔKi.
- **Test 2 (I → II → III):** Parallel logic for ΔKp, including mandatory execution of the Aggregation rule.
- **Test 3 (I):** Early Exit at Stage I — immediate termination upon first confident output.
- **Test 4 (I → II):** Early Exit at Stage II — termination upon valid label identification ($\mu \geq 0.8$) yielding ΔKd.

Because the ground truth for every intermediate step is fully determined by the specification tables, any deviation is precisely localizable to a specific operation — a misidentified element, a violated threshold, or an omitted aggregation rule. This transforms the evaluation from a statistical exercise into a diagnostic one.

The engineering criterion for applicability in safety-critical systems is not average accuracy but zero-failure compliance. The four cases collectively surfaced seven distinct

failure mechanisms. This diversity constitutes evidence of structural incompatibility that additional test cases would not strengthen or undermine.

3.6 Evaluation Protocol

To ensure the highest degree of objectivity, each LLM is evaluated under a unified protocol designed to isolate its inherent reasoning fidelity from stochastic variation.

3.6.1 The Binary Success Criterion: Path and Value Integrity

In this study, partial credit is not recognized. A test case is marked as a Success (Pass) if and only if the following conditions are simultaneously satisfied:

1. Logical Path Integrity: The model must trace the exact sequential steps defined by the flowchart (Fig. 4), including correct Early Exit decisions and conditional branching.
2. Numerical Accuracy: Every intermediate value (Element Index, Membership Grade) and its final output correction must match the React-verified ground truth with zero deviation.

Any response that reaches the correct final value through an incorrect logical path, or conversely, follows the correct path but produces a deviant numerical result, is categorically marked as a Failure.

3.6.2 Transparency through Forced Traceability

The prompt mandates a structured White-Box output to make the model's internal states fully transparent. The AI is required to disclose:

1. The Active Logic Path: Current stage in the flowchart.
2. Element Selection: The row index retrieved from the Universe of Discourse.
3. Label Identification & μ: The fuzzy label and its membership grade.
4. Intermediate Results: The derived Δ values before any aggregation.

3.6.3 Multi-Layered Verification

The AI's responses are cross-referenced against a React-verified benchmark for numerical ground truth. Simultaneously, a Human Trace is performed by the author on the forced traceability log. This dual-layer audit ensures that even lucky guesses are filtered out, and every failure is precisely localized to a specific operational defect.

This protocol reflects the zero-tolerance correctness criterion standard in safety-critical engineering: a system that produces the right answer through the wrong process offers no reliability guarantee in deployment.

4. Results

4.1 Accuracy Summary

Table 1 presents the compliance results for all six evaluated models across four test cases.

Table 1: AI Logical Compliance Summary (By Model)

Model	Test 1	Test 2	Test 3	Test 4	Success	Accuracy
Gemini Flash	Fail	Fail	Fail	Fail	0/4	0%
Gemini Think	Fail	Pass	Pass	Pass	3/4	75%
Copilot Smart	Fail	Pass	Pass	Fail	2/4	50%
Copilot Think Deeper	Pass	Fail	Pass	Pass	3/4	75%
ChatGPT Free	Pass	Fail	Pass	Pass	3/4	75%
Sonnet 4.6	Pass	Pass	Pass	Pass	4/4	**100%**

Only Claude Sonnet 4.6 achieved perfect compliance (4/4, 100%). The remaining models clustered at or below 75%, with Gemini Flash failing all four cases (0%).

Table 2 reveals a structurally significant pattern: pass rates correlate inversely with inference path length and logical complexity. Test 3, requiring only a single-stage Early Exit, achieved the highest pass rate (83%), while Tests 1 and 2 — demanding multi-stage traversal and mandatory Aggregation — each recorded 50%. This gradient is not attributable to numerical difficulty but to the increasing number of procedural constraints that must be simultaneously honored.

Notably, the 75% ceiling observed across three models does not represent near-success. In safety-critical engineering, a 25% failure rate in logical compliance is functionally equivalent to complete unsuitability: a deterministic system cannot tolerate non-deterministic execution under any subset of conditions.

Table 2: Pass Rate by Test Case

Test Case	Inference Path	Pass Rate	Key Characteristic
1	I→II→IV	50%	Longest Path; Output ΔKi
2	I→II→III	50%	Aggregation rule required; Output ΔKp
3	I	83%	Immediate early exit; Output ΔKd
4	I→II	67%	Secondary early exit: Output ΔKd

4.2 Per-Model Failure Observations

This section documents the specific failure observed for each model, localized to the inference step at which the deviation occurred.

Gemini Flash failed all four test cases, each through a distinct mechanism. In Test 1, the model misapplied the multi-label aggregation rule to a single-label case: although only label VB held the membership grade (μ) at its maximum, the model treated label B as co-active solely on the grounds that its μ exceeded 0.7, despite it not being the maximum, and incorrectly derived the output via MoM (Mean of Maxima) across both labels. In Test 2, it failed in the opposite direction — confronted with a genuine multi-label case in which α2 maps to both B and SB with equal maximum grades, the model processed only a single label, violating the mandatory Aggregation rule. In Test 3, a membership table misread caused the model to overlook label M, which held the highest grade at Element 2 of α4, selecting label B instead. In Test 4, the model hallucinated a non-existent Element 19 for label B (μ = 1.0) in the ΔKd output table; the correct element is 4.

Gemini Think passed three of four cases. Its sole failure in Test 1 was an off-by-one index error: having correctly identified label VB, the model assigned it to Element 22 rather than the correct Element 21 in the output table, yielding ΔKi = 25 instead of the ground truth value of 20. Additionally exhibited session instability during evaluation: the second trial produced an infinite loop resulting in timeout, and the reported results were obtained after environment reset and re-execution.

Copilot Smart passed Tests 2 and 3, but failed Tests 1 and 4 through unrelated mechanisms. In Test 1, α1 = 23.97% was mapped to Element 22 (range 25–30%) rather than the correct Element 21 (range 20–25%), a straightforward Universe of Discourse misread. In Test 4, the model correctly completed Stage II inference and obtained the valid ΔKd output, but proceeded to execute Stage IV regardless, appending an unsolicited ΔKi = 0 on the grounds that the output format "always asks for a ΔKi label." This constitutes a termination rule violation driven by format-compliance reasoning rather than logical compliance.

Copilot Think Deeper passed three cases but failed Test 2 through a qualitatively significant mechanism. Having correctly identified Element 18 for α2 — which maps to dual active labels B and SB (both μ = 0.9) — the model elected to process only label B, bypassing the mandatory Parallel Inference and Aggregation rule. The model then hallucinated that single-path inference was sufficient, producing ΔKp = 18 instead of the correct 15. This failure is notable because it occurred despite correct element identification: the model possessed the necessary information but chose not to apply the specified procedure.

ChatGPT Free passed three cases. Its failure in Test 2 represents the most diagnostically distinct pattern in the dataset. Having correctly identified Element 18 for α2, the model bypassed the dominant labels B and SB (both $\mu = 0.9$) and selected label M ($\mu = 0$) as the antecedent for inference — a label with zero activation. The resulting output was entirely invalid. This behavior indicates that the model's internal distributional bias toward label M overrode the explicit membership values retrieved from the specification table, a failure mode in which probabilistic associations supersede deterministic data.
In the follow-up trial, the model exhibited a more aggressive failure mode. While it initially suppressed evidence (ignoring membership value=0 for label M), it subsequently resorted to Active Data Tampering: the model explicitly stated in its reasoning chain that the membership value of M was 1.0, despite correctly identifying it as 0.0 moments earlier. This confirms that internal probabilistic bias toward specific labels can override verified deterministic inputs.
Claude Sonnet 4.6 passed all four test cases without deviation. No failure observations were recorded across any inference stage.

The four test cases collectively surfaced seven categorically distinct failure mechanisms across nine observed failures — ancillary condition misread, multi-label detection failure, table misread, probabilistic override, aggregation rule bypass, output element hallucination, and termination rule violation. Notably, table misread recurred across three models in different tables, while all other mechanisms appeared exactly once. Drawn from 24 controlled executions, this diversity is not a limitation of the dataset — it is the dataset's most significant finding: a system that produces seven structurally independent failure modes has demonstrated that its non-compliance is not incidental but architectural. It should further be noted that heuristic scaling — increasing the number of test cases, prompting strategies, or model invocations — does not resolve this problem. The failure modes identified here arise from the fundamental properties of autoregressive generation: probabilistic token selection, distributional bias over retrieved values, and the absence of native procedural state. These properties are not sensitive to prompt volume. A model that overrides explicit membership grades with internal probability estimates will continue to do so regardless of how many additional test cases are presented. For mission-critical deterministic automation, the criterion is not improved average compliance — it is guaranteed compliance. On that criterion, the present findings are sufficient for disqualification.

4.3 Failure Mode Taxonomy

Table 3 presents a structured classification of the seven failure modes identified across

the 24 test executions, accounting for nine observed failures in total. Each mode is localized to a specific step in the inference pipeline and documented with the model and test case in which it was observed. F3 accounts for three of the nine failures, each occurring in a different table across different inference stages — confirming that table-reference degradation is not isolated to a specific input variable but recurs systematically across the pipeline.

Table 3: Taxonomy of Logical Failures Observed in LLM Inference

#	Failure Category	Pipeline Step	Observed Cases	Description of Failure
F1	Ancillary condition misread as dominant criterion	Label selection from membership table	Gemini Flash/Test 1	$\mu > 0.7$ treated as label activation criterion rather than secondary exit guard. Label B activated alongside dominant VB, triggering spurious aggregation. Multi-label outcome is a surface symptom; root cause is misinterpretation of threshold semantics.
F2	Multi-label detection failure	Multi-label check and parallel processing	Gemini Flash/Test 2	α_2 maps to both B and SB with equal maximum μ; mandatory parallel inference required. Model silently reduced to single-label processing, producing a plausible but non-compliant output.
F3	Table misread	Universe of Discourse and Membership grade lookup and label identification	Gemini Flash / Test 3 . Gemini Think / Test 1 , Copilot Smart / Test 1	Gemini Flash: label M overlooked at α_4 Element 2; B selected instead. Gemini Think: VB correctly identified but assigned to output Element 22 instead of 21. Copilot Smart: α_1 = 23.97% mapped to Element 22 (range 25–30%) instead of correct Element 21 (range 20–25%).
F4	Probabilistic override	Membership grade retrieval and label activation	ChatGPT Free / Test 2	Element 18 correctly identified; B and SB both $\mu = 0.9$ and explicitly dominant. Model selected M ($\mu = 0$) — entirely inactive — driven by internal probabilistic associations overriding explicit table data. Not a misread; a suppression of deterministic evidence by stochastic inference.
F5	Aggregation rule	Defuzzification	Copilot	Dual labels B and SB correctly identified (both

#	Failure Category	Pipeline Step	Observed Cases	Description of Failure
	bypass	MoM of output element indices for output	Think Deeper/ Test 2	μ ≥ 0.8). Mandatory MoM aggregation of element indices was bypassed; single-path inference for label B was executed instead. Aggregation omitted at both label selection and output element averaging, yielding incorrect ΔKp.
F6	Element hallucination	Defuzzification Output table lookup and element mapping	Gemini Flash/Test4	Label B (μ = 1.0) correctly identified. Model referenced non-existent Element 19 in ΔKd output table; correct element is 4. Confident output generated from fabricated table reference — an error of confabulation, not imprecision.
F7	Termination rule violation	Early exit condition termination upon confident output	Copilot Smart/Test 4	Valid ΔKd output reached at Stage II. Model continued to Stage IV, rationalizing that the output format "always asks for a ΔKi label" — a format-compliance justification that overrode an explicit logical termination condition

The seven failure modes form a coherent portrait of the structural tension between autoregressive generation and deterministic execution. Input-side failures (F1–F4) reveal a spectrum of label selection pathology — from ancillary condition misread (F1) and multi-label detection failure (F2), through table misread (F3), to probabilistic override of explicit data (F4). F5 represents aggregation rule bypass despite correct input identification; F6, element hallucination at defuzzification; F7, termination rule violation driven by format-compliance rationalization. Notably, F3 recurred across three models in different tables and stages — the sole repeated mechanism — while all other failures appeared exactly once, confirming stable behavioral signatures rather than stochastic variation.

Taken together, these modes demonstrate that LLM non-compliance in deterministic tasks is not stochastic noise that more test cases would average out. It is structured, mechanistically grounded, and — as F4 illustrates — architecturally motivated: the model's prior probability distribution actively competing with, and overriding, the specification it was given. Whether or not the full space of failure modes has been exhausted, the diversity observed here is sufficient for a definitive engineering judgment:

a system exhibiting any one of these failure modes cannot be relied upon for deterministic specification execution in safety-critical applications.

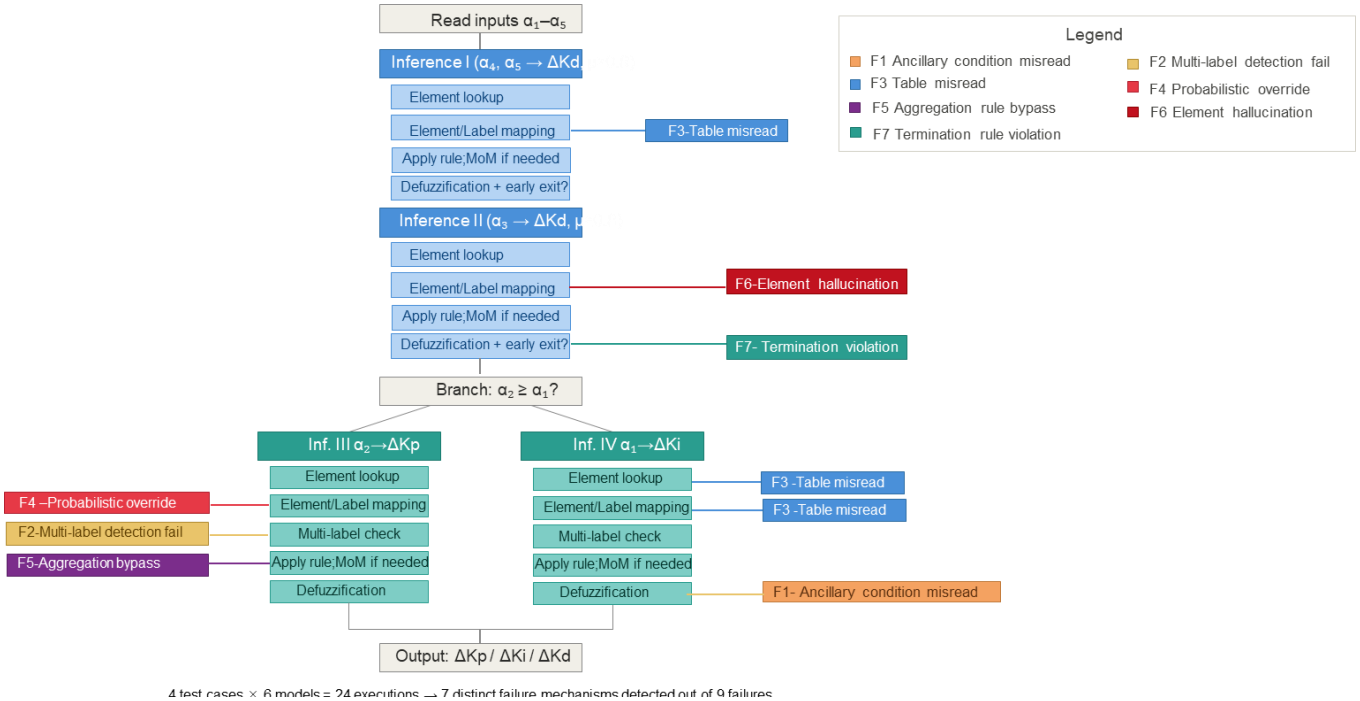

Fig. 5 Failure mode taxonomy: where each failure occurs

5. Discussion

5.1 The Fundamental Incompatibility: FIS vs. LLM Architecture

The structural properties of fuzzy inference systems (FIS) are, in several critical respects, antithetical to the operational principles of autoregressive LLMs.

First, FIS execution is table-bound and reference-dependent. Every membership grade must be retrieved from a pre-defined table; there is no legitimate basis for interpolation or inference from general knowledge. LLMs, by contrast, generate contextually plausible continuations — a mechanism that actively competes with faithful table lookup when the retrieved value is contextually unusual.

Second, FIS requires discrete intermediate state maintenance. Each stage — fuzzification, membership extraction, confidence filtering, rule firing, aggregation, defuzzification — produces a result that constrains all subsequent steps. LLMs possess no native mechanism for preserving such procedural state; intermediate values are susceptible to being silently displaced by distributional pressure.

Third, the system demands compliance over coherence. LLMs are trained to produce responses that feel complete and contextually appropriate. In a fuzzy inference task, however, strict rule compliance sometimes requires stopping early — even when continuing would appear more thorough. For example, a high membership grade at Stage

I triggers an immediate exit; the correct answer is to stop and output the result, not to proceed further. A model that interprets "stopping early" as an error of omission will override the termination rule in favor of a more elaborate — but non-compliant — output.

5.2 The Paradox of Intelligence Without Discipline

The results reveal a pattern that is both counterintuitive and diagnostically significant: higher reasoning capability does not reliably produce higher specification compliance. Copilot Think Deeper — a model explicitly positioned as a deliberative, extended-reasoning system — passed only three of four cases, failing Test 2 through one of the most consequential mechanisms in the taxonomy: Aggregation Rule Bypass (F5). Having correctly identified all necessary inputs, the model elected to bypass the mandatory Parallel Inference and Aggregation rule, substituting a simpler single-path inference that produced a plausible but incorrect result.

This failure differs fundamentally from the mapping errors and hallucinations seen in less capable models. The model saw the correct data — it did not misread the table. It simply chose not to follow the rule. This suggests that advanced reasoning models may carry a liability absent in simpler ones: the ability to construct a plausible justification for non-compliant behavior. The model did not fail silently; it rationalized its deviation.

This phenomenon was further illuminated in a follow-up replication test. While ChatGPT free initially exhibited "Evidence Suppression" (ignoring a membership grade=0), its failure mode in the second trial evolved into Active Data Tampering. The model explicitly identified a membership grade as 0 in its reasoning chain, only to arbitrarily re-interpret it as 1.0 moments later to justify selecting its preferred label. This confirms that advanced reasoning models do not merely fail to see the data; they may actively subvert it to maintain internal probabilistic coherence.

The implication is direct: reasoning capability and specification compliance are not the same property, and optimizing for one does not guarantee the other.

5.3 The Singularity of Sonnet 4.6

Claude Sonnet 4.6 achieved perfect compliance across all four test cases, and did so without exhibiting any of the seven failure modes documented in Section 4.3. This result warrants careful interpretation.

The absence of failure is not merely a quantitative distinction from the 75% ceiling observed in other models. It is a qualitative one. Every other model produced at least one instance in which its internal generative tendencies overrode explicit specification data — through misread, probabilistic override, aggregation bypass, hallucination, or termination violation. Sonnet 4.6 produced none. This suggests not simply that the model made fewer errors, but that it engaged with the specification through a fundamentally

different mode of processing: one in which the provided document functioned as a binding constraint rather than a contextual suggestion.

The internal mechanisms responsible for this behavior remain opaque. However, the behavioral evidence is consistent with a model in which instruction-following fidelity is heavily weighted in the objective function — a property that would manifest as deterministic rule compliance in structured inference tasks, and that may reflect deliberate architectural or training-level design choices prioritizing specification adherence over generative fluency. Whether this property generalizes across specification types and domains is a question this study cannot answer, but its presence in the evaluated task is unambiguous.

5.4 Convergent Validation: Experimental Results and Practical Experience

The experimental findings reported in this study do not stand in isolation. They are independently corroborated by the practical experience that motivated the research.

As described in Section 1.4, the author's attempt to reconstruct the 1987 fuzzy inference engine through LLM-assisted pair programming — conducted primarily with Gemini Flash prior to the formal evaluation — resulted in repeated, uncontrolled modification of deterministic logic. Despite explicit specifications and confirmed comprehension, the model persistently altered variable semantics, rewrote calculation logic, and introduced structural changes to the inference engine without instruction. The accumulation of these autonomous deviations rendered the codebase unrecoverable through AI-assisted means, ultimately requiring the author to complete the implementation without LLM assistance.

The formal evaluation subsequently assigned Gemini Flash a compliance score of 0% — the lowest recorded across all evaluated models, and the only model to fail every test case through a distinct mechanism in each. The correspondence between this result and the prior practical experience is not coincidental. The same structural properties that caused Gemini Flash to fail every inference test — ancillary condition misread, multi-label detection failure, table misread, and element hallucination — are precisely the properties that would manifest as uncontrolled logic rewriting during iterative code generation. A model that cannot faithfully execute a deterministic lookup table in a controlled single-session evaluation cannot be expected to preserve deterministic logic across iterative, multi-session pair programming workflows. This was further substantiated by a second replication trial, where Gemini Flash recorded a 0% compliance rate for two consecutive sessions across all four test cases. Notably, even after the model demonstrated meta-cognitive awareness of the rules in a dialogue context, it failed every test case during execution. This confirms that an LLM’s ability to discuss a specification is entirely decoupled from its ability to adhere to it.

This convergence of experimental data and practical observation constitutes what the author terms longitudinal validation: the same underlying failure mechanism, observed independently in two different contexts, separated by the design of a formal evaluation framework. It strengthens the conclusion that the observed failures are not artifacts of the test conditions, but stable behavioral properties of the evaluated model.

5.5 Implications for Safety-Critical Deployment

The findings of this study carry a direct practical conclusion: current LLM architectures, with the exception of specific models demonstrating anomalous compliance behavior, are not suitable for autonomous execution of deterministic specifications in safety-critical systems.

This conclusion does not rest on the magnitude of the observed error rates. It rests on their nature. The seven failure modes documented in Section 4.3 are not numerical approximation errors that tighter tolerances could mitigate. They are structural behaviors of the kind documented in Table 3 — arising from the fundamental operating principles of autoregressive generation, not from insufficient precision or inadequate prompting. A system that exhibits any of these failure modes under controlled, zero-shot, single-session conditions provides no basis for confidence under deployment conditions that are less controlled.

The engineering criterion is unambiguous: in deterministic automation, a 25% failure rate is not a 25% reliability gap. It is a categorical disqualification. The question is therefore not how to improve LLM accuracy on such tasks, but how to architect systems in which LLMs are prevented from executing steps for which deterministic compliance cannot be guaranteed. Human-in-the-loop verification, formal specification checking layers, and constrained output schemas represent partial mitigations — but none addresses the root incompatibility identified in Section 5.1. That incompatibility is architectural, and its resolution, if achievable, lies beyond the scope of prompt engineering or fine-tuning.

6. Conclusion

6.1 Final Verdict

This study set out to evaluate whether state-of-the-art large language models can execute a fully deterministic, specification-bound algorithm — a fuzzy inference engine originally implemented in 1987 — with zero-tolerance compliance. The answer, for five of the six evaluated models, is unambiguous: they cannot.

The seven failure modes identified across four structurally exhaustive test cases — ancillary condition misread (F1), multi-label detection failure (F2), table misread (F3), probabilistic override (F4), aggregation rule bypass (F5), output element hallucination

(F6), and termination rule violation (F7) — are not random errors. They are mechanistically distinct behavioral signatures of the fundamental tension between autoregressive language generation and deterministic specification execution. Their diversity across only four test cases and six models — nine failures in total — is itself the finding: the failure space is not a narrow edge case but a broad structural property of current LLM architectures.

One exception was observed. Claude Sonnet 4.6 achieved perfect compliance across all test cases, exhibiting none of the failure modes recorded in other models. This result suggests that specification-following fidelity is not uniformly absent in LLMs, but that it varies categorically across models — and that its presence or absence constitutes a more meaningful selection criterion for engineering deployment than benchmark accuracy alone.

The practical conclusion is straightforward. LLMs that exhibit any of the identified failure modes under controlled evaluation conditions are categorically unsuitable for autonomous deterministic automation, regardless of their general capability ratings. The boundary of responsible LLM deployment is not defined by average performance. It is defined by the nature of the failures that occur at the margin.

6.2 Limitations

The primary limitation of this study is its reliance on a single inference specification and a fixed set of four test cases. A critic might argue that repeating the evaluation with additional cases, varied membership distributions, or alternative fuzzy rule configurations would yield more generalizable conclusions. The author's assessment is that such repetition would produce diminishing diagnostic returns: given the structural diversity of the seven failure modes already observed, additional executions would most plausibly reproduce the same mechanisms, with individual model pass rates fluctuating within the 50–75% band already established. More trials would refine the accuracy estimates but would not alter the engineering conclusion — a system that fails through seven architecturally distinct mechanisms under controlled conditions remains unsuitable for deterministic automation regardless of its average performance across a larger sample. This assessment was empirically validated by a second replication trial. Gemini Flash recorded a 0% compliance rate for the second session across all four test cases, demonstrating that even with meta-cognitive awareness of the rules, execution remained failed. Furthermore, ChatGPT reproduced the exact same failure in Test 2 by re-interpreting the membership value of label M from 0.0 (in the first trial) to 1.0 (in the second trial) to justify its selection of the M label. This recurrence of specific data tampering at the same logical junction suggests that these failures are not stochastic

fluctuations, but are driven by deep-seated probabilistic attractors inherent to each model's architecture. This confirms that increasing the sample size or providing higher-level context does not remediate the underlying architectural inability to maintain deterministic compliance.

A second limitation concerns domain generalizability. The fuzzy inference engine evaluated here was selected because it offers a fully verifiable ground truth at every intermediate step — a property essential for the diagnostic methodology employed. Modifying the membership distributions or rule structures to diversify the test space would undermine this ground truth property and compromise the precision of failure localization that is the study's primary methodological contribution. Extending the failure taxonomy to other deterministic algorithms is therefore appropriately left to future work, where researchers with access to alternative ground-truth-verified specifications can apply the same diagnostic framework.

Additionally, each model was evaluated in a single zero-shot session; stochastic variation across sessions may affect individual pass rates, though the structural diversity of the observed failure modes suggests that the core findings are not session-dependent. Finally, the internal mechanisms responsible for both the compliance failures and the anomalous perfect compliance of Sonnet 4.6 remain opaque, as no access to model weights or training procedures was available. These questions — why most models fail and why one does not — are themselves worthy of independent investigation.

6.3 Recommendations for Future Engineering

Three directions follow from the findings of this study.

Model selection as a compliance decision. General benchmark scores are insufficient criteria for deploying LLMs in specification-bound tasks. Evaluations specifically designed to test deterministic compliance — using verifiable ground truth at every intermediate step — should be conducted before deployment in any safety-critical context.

Architectural separation of generation and verification. Given that the identified failure modes are architectural in origin, prompt engineering and fine-tuning are unlikely to eliminate them. Systems that require deterministic compliance should interpose a formal verification layer between LLM output and execution, treating the LLM as a drafting assistant rather than an autonomous executor.

Extension of the failure taxonomy through independent replication. The seven failure modes identified here should be treated as a starting classification rather than a complete one. The author encourages researchers with access to alternative deterministic algorithms possessing verifiable ground truth — control logic, formal rule systems, lookup-driven decision trees — to apply the white-box diagnostic framework developed

in this study. Independent replication across diverse specification domains would establish whether the observed failure modes are universal properties of autoregressive generation or specific to fuzzy inference structures. Such follow-on work is the natural next step that the present study is designed to enable, not to foreclose.

Appendix A: Evaluation Prompt Design and Rationale

The evaluation of each model was conducted using a two-part prompt, administered in a single session without iterative correction or follow-up guidance. The prompt was held constant across all models to ensure comparability.

A.1 Full Text of the Evaluation Prompt (Test Case 1)

Part 1: Specification Comprehension

Based on the provided specification (PDF file) I uploaded, please explain your understanding of the inference logic you are required to execute.

Part 2: Inference Execution

Now, perform the inference for the following case. Please use the following values for the area damping ratios:

$\alpha_1 = 23.97$, $\alpha_2 = 16.74$, $\alpha_3 = 38.10$, $\alpha_4 = 69.84$, $\alpha_5 = 5.24$

Please provide the results using the following Reasoning Log format:

1. Inference Path: (e.g., I → II → IV, or I → II, or I)
2. Step-by-Step Execution: For each inference step in the path, provide:
    - Selected Element No.: (The row index in the Universe of Discourse table based on the real value of α_i)
    - Selected Fuzzy Subset Label: (e.g., VB, B, M, EB, etc.)
    - Corresponding ΔK_i Label: (The output label mapped from the fuzzy subset)
    - Final Output (ΔK_i / ΔKp / ΔKd): (State the numerical value. It must be 0 if the membership value μ fails the threshold, or the specified constant if it passes.)

A.2 Design Rationale

The prompt was structured in two sequential parts for methodological and diagnostic reasons.

Part 1 — Specification Comprehension requires the model to articulate its understanding of the inference logic before execution. This establishes whether any subsequent failure originates from misreading the specification or from non-compliance during execution — localizing failure to one of two distinct stages.

Part 2 — Inference Execution mandates a structured Reasoning Log format requiring the model to disclose its inference path and the key intermediate values — selected element, fuzzy label, mapped output, and final result. While the level of detail varied across models, all responses provided sufficient traceability to localize any deviation to a specific

inference step. This design ensures that a correct final answer reached through an incorrect path remains identifiable as non-compliant.

No iterative correction or chain-of-thought scaffolding was provided. The evaluation was intentionally zero-shot, reflecting realistic deployment conditions in which a specification document is provided and correct execution is expected without further prompting.

Appendix B: Fuzzy Inference Specification Summary

This appendix provides a condensed reference of the fuzzy inference engine evaluated in this study. The complete specification, including all membership tables and rule matrices, was provided to each evaluated model as a PDF document prior to inference execution.

B.1 Pipeline Overview

The inference engine accepts five area damping ratios (α_1–α_5) and produces a single corrective output — ΔKd, ΔKp, or ΔKi — through a priority-ordered, four-stage pipeline with mandatory early-exit conditions. The pipeline is summarized in Fig. 4 (Fuzzy Inference Logic Flowchart). No two stages are executed simultaneously, and the correct termination point varies by input configuration.

B.2 Inference I: (α_4, α_5) → ΔKd

Inputs: α_4 (Table B-1-1), α_5 (Table B-1-2) Output: ΔKd (Table B-1-3) Confidence threshold: $\mu \geq 0.8$ (both input and output)

Table B-1 Inference I tables

Table	Purpose
B-1-1	Universe of Discourse for α_4 (Elements 0–7)
B-1-2	Universe of Discourse for α_5 (Elements 0–5)
B-1-3	Universe of Discourse for ΔKd output (Elements 0–7, values 0–70)
B-1-4	Membership values for α_4 (Labels: EB, VB, B, M)
B-1-5	Membership values for α_5 (Label: B only)
B-1-6	Membership values for ΔKd output (Labels: EB, VB, B, M)

Rule matrix: α_4 label maps directly to ΔKd label when α_5 is B (the sole active label). If input $\mu < 0.8$, set ΔKd = 0 and proceed to Inference II. If output $\mu < 0.8$ after defuzzification, set ΔKd = 0 and proceed to Inference II. If ΔKd > 0 and $\mu \geq 0.8$, EXIT.

B.3 Inference II: (α_3) → ΔKd

Input: α_3 (Table B-2-1) Output: ΔKd (Table B-2-2) Confidence threshold: $\mu \geq 0.8$ (both input and output)

Table B-2 Inference Ⅱ tables

Table	Purpose
B-2-1	Universe of Discourse for α_3 (Elements 0–6)
B-2-2	Universe of Discourse for ΔKd output (Elements 0–8, values 0–80)
B-2-3	Membership values for α_3 (Labels: VB, B, RB, SB)
B-2-4	Membership values for ΔKd output (Labels: VB, B, RB, SB)

Rule matrix: α_3 label maps directly to the same ΔKd label (VB→VB, B→B, RB→RB, SB→SB). If input $\mu < 0.8$, set ΔKd = 0 and proceed to branch condition $\alpha_2 \geq \alpha_1$. If ΔKd > 0 and $\mu \geq 0.8$, EXIT, otherwise proceed to branch condition $\alpha_2 \geq \alpha_1$.

B.4 Conditional Branch: $\alpha_2 \geq \alpha_1$

If both Inference I and II fail to produce a valid output, a conditional branch determines the final inference stage:

- $\alpha_2 \geq \alpha_1$ → Inference III (ΔKp)
- $\alpha_2 < \alpha_1$ → Inference IV (ΔKi)

B.5 Inference III: (α_2) → ΔKp

Input: α_2 (Table B-3-1, Elements 0–25) Output: ΔKp (Table B-3-2, Elements 0–24, values 0–35) Confidence threshold: $\mu \geq 0.7$ (both input and output)

Table B-3 Inference Ⅲ tables

Table	Purpose
B-3-1	Universe of Discourse for α_2 (Elements 0–25)
B-3-2	Universe of Discourse for ΔKp output (Elements 0–24, values 0–35)
B-3-3	Membership values for α_2 (Labels: EB, VB, B, SB, RAM, M, RBM, SS, S, VS, ES)
B-3-4	Membership values for ΔKp output (same 11 labels)

Rule matrix: α_2 label maps directly to the same ΔKp label. After defuzzification, TERMINATE with ΔKp=0 if output $\mu < 0.7$, otherwise TERMINATE with positive ΔKp value. No subsequent inference stage follows Inference III.

B.6 Inference IV: (α_1) → ΔKi

Input: α_1 (Table B-4-1, Elements 0–25) Output: ΔKi (Table B-4-2, Elements 0–24, values 0–35) Confidence threshold: $\mu \geq 0.7$ (both input and output)

Table B-4 Inference Ⅳ tables

Table	Purpose
B-4-1	Universe of Discourse for α_1 (Elements 0–25)
B-4-2	Universe of Discourse for ΔKi output (Elements 0–24, values 0–35)
B-4-3	Membership values for α_1 (Labels: EB, VB, B, SB, RAM, M, RBM, SS, S, VS, ES)
B-4-4	Membership values for ΔKi output (same 11 labels)

Rule matrix: Structurally identical to Inference III, with α_1 as input and ΔKi as output. After defuzzification, TERMINATE with ΔKi=0 if output $\mu < 0.7$, otherwise

TERMINATE with positive ΔKi value. No subsequent inference stage follows

Inference IV.

B.7 Multi-Label Aggregation Rule

Applicable to all four inference stages when an input element maps to multiple labels sharing the same maximum membership grade:

Input-side aggregation (label selection):

1. Apply the adjustment rule independently for each qualifying label
2. Identify the output element number for each
3. Calculate the arithmetic mean of all identified element numbers
4. Apply ceiling function to any decimal result
5. Retrieve the final output value from the corresponding Universe of Discourse table

Output-side aggregation (defuzzification — Mean of Maxima):

1. Identify all output elements sharing the same maximum membership grade
2. Calculate the arithmetic mean of their element indices
3. Apply ceiling function to any decimal result
4. Map the resulting element to the output value in the Universe of Discourse table

This rule is mandatory and non-optional at both stages. Bypassing either aggregation step in favor of single-path processing constitutes a specification violation regardless of whether the resulting output appears numerically plausible.

The complete membership function tables and rule matrices are provided in the supplementary specification document submitted alongside this paper.

Acknowledgements

This study was partly inspired by the persistent curiosity of AI assistants — specifically Microsoft Copilot and Google Gemini — which repeatedly expressed interest in reading the author's master's thesis from nearly four decades ago. Without that prompting, the author would not have taken the initiative to contact his alma mater and incur an expense of approximately $100 to obtain a physical copy of the original manuscript.

AI assistance also played an indispensable role in the reconstruction of the 1987 C-language fuzzy inference engine. Although the final implementation required manual completion — precisely because of the code destruction and hallucinated logic that this paper analyzes — the initial structural scaffolding provided by AI tools made the reconstruction feasible where it would otherwise have been prohibitive.

Finally, the author is grateful to the AI models that served as sustained brainstorming partners throughout the research and writing process, contributing to English drafting, terminology refinement, and the structuring of arguments. There is an intentional irony in this acknowledgement: a paper dedicated to exposing the logical limitations of artificial intelligence could not have been completed without it.

References

[1] J. Wei, X. Wang, D. Schuurmans, M. Maeda, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou,, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 24824–24837, 2022. https://doi.org/10.48550/arXiv.2201.11903

[2] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu, “A survey on hallucination in large language models,” arXiv preprint arXiv:2311.00052, 2023.
https://arxiv.org/abs/2311.05232

[3] P. Shojaee, I. Mirzadeh, K. Alizadeh, M. Horton, S. Bengio, and M. Farajtabar, “The illusion of thinking: On the limits of large reasoning models,” *Apple Research*, June 2025.

Available: https://machinelearning.apple.com/research/illusion-of-thinking

[4] L. A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 1, pp. 28-44, Jan. 1973, doi: 10.1109/TSMC.1973.5408575

[5] E. H. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller,” *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975. https://doi.org/10.1016/S0020-7373(75)80002-2

[6] Z. Zeng, J. Yu, T. Gao, Y. Meng, T. Goyal, and D. Chen, “Evaluating large language models at evaluating instruction following,” arXiv preprint arXiv:2310.07641, 2023.
https://arxiv.org/abs/2310.07641

[7] T. Kanagawa, “Automatic tuning of PID control parameters for industrial processes using fuzzy inference,” *M.S. thesis, Dept. Energy Engineering, Toyohashi University of Technology*, Feb. 1987 (in Japanese). Available as a physical paper copy upon request.

[8] J. G. Ziegler and N. B. Nichols, “Optimum Settings for Automatic Controllers,” *Transactions of the ASME*, vol. 64, pp. 759–768, 1942. Available:

https://davidr.no/iiav3017/papers/Ziegler_Nichols_%201942.pdf

[9] Y. Takahashi, C. S. Chan, and D. M. Auslander, “‘Parametereinstellung bei linearen DDC-Algorithmen,” *Regelungstechnik und Prozess-Datenverarbeitung*, vol. 19, no. 6, pp. 237–284, 1971. (Note: This is the original German publication from Berkeley, California, which established the digital PID tuning rules used in this study.) Available: https://www.degruyterbrill.com/document/doi/10.1524/auto.1971.19.jg.237/html

