Technical Report for AI4Math-2026 Track 1: Automated Semantic Alignment Verification and Error Categorization of Lean 4 Formalizations via Decomposition-Guided Auditing

Huan Vu [1 2] **Tien Cuong Nguyen** [2] **Thien Van Luong** [1 2]

Abstract

We present a semantic alignment verification pipeline for Lean 4 autoformalization developed for the AI4Math @ ICML 2026 FormalRx Challenge. Given an informal mathematical statement and a candidate Lean formalization, the system predicts semantic alignment, generates corrected formal statements, localizes erroneous segments, and classifies errors using the 28-category SCI taxonomy. Our approach combines structured decomposition of informal mathematics, LLM-based semantic auditing, correction generation, and a hybrid SCI classifier integrating deterministic token substitution rules with model-based classification. Experiments on the official FormalRx benchmark show that our best system achieves an Overall score of 0.3397, with strong Verdict F1 (0.7327) and Correction accuracy (0.7671). Results indicate that tightly coupled semantic reasoning and correction generation outperform heavily decomposed multi-stage pipelines, while symbolic heuristics provide effective complementary signals for SCI categorization.

1. Introduction

Recent progress in large language models (LLMs) has accelerated research in autoformalization: the task of translating informal mathematical language into formally verified statements in proof assistants such as Lean 4. Reliable autoformalization could substantially improve theorem proving, formal verification, mathematical knowledge management, and AI-assisted scientific reasoning. However, evaluating whether a generated formal statement faithfully preserves the meaning of the original natural-language mathematics remains a major open challenge.

Existing evaluation approaches often reduce semantic fidelity to a binary correctness label or a scalar similarity score. Such evaluations provide limited diagnostic information and fail to identify how a formalization diverges from the intended mathematical meaning. In practice, semantic misalignment can arise from many distinct sources, including incorrect quantifiers, weakened hypotheses, type mismatches, omitted constraints, altered conclusions, or implementation-level formalization mistakes. These errors are structurally diverse and require fine-grained analysis beyond coarse correctness judgments.

The Semantic Alignment Evaluation for Autoformalization track of the 3rd AI4Math @ ICML 2026 Challenge (ai4, 2026) addresses this problem through the FormalRx benchmark and the SCI error taxonomy (for, 2025), which decomposes semantic alignment failures into fine-grained diagnostic categories. Given an informal mathematical statement and a candidate Lean 4 formalization, participants must determine whether the pair is semantically aligned and, if misaligned, identify the error category, localize the erroneous segment, and generate a corrected formal statement.

In this work, we propose a structured semantic auditing pipeline for semantic alignment evaluation. Rather than directly comparing natural language against Lean syntax in an end-to-end manner, our method first converts the informal statement into a lightweight intermediate semantic representation consisting of object types, assumptions, and proof goals. This decomposition reduces linguistic ambiguity and partially bridges the representational gap between compressed natural-language mathematics and structurally explicit formal statements.

Using this intermediate representation, our system performs semantic auditing to jointly predict alignment verdicts and generate minimally modified corrected formalizations. We further introduce a hybrid symbolic–LLM error classifier that combines deterministic rule-based detection for low-entropy symbolic edits with model-based classification for higher-level semantic discrepancies. Error localization is derived automatically through token-level differencing be-

[1]Business AI Lab, National Economics University, Hanoi, Vietnam [2]VNPT AI, Hanoi, Vietnam. Correspondence to: Huan Vu <huanv@neu.edu.vn>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

tween the original and corrected formal statements.

Our approach is designed to be modular, interpretable, and robust in large-scale evaluation settings. The pipeline includes structured fallback mechanisms, retry handling, concurrent processing, and resumable execution for efficient inference over the FormalRx benchmark (for, 2025). While the system is fully LLM-driven at the semantic reasoning level, decomposition and correction generation provide structured intermediate signals that improve both interpretability and downstream SCI categorization.

The remainder of this report describes the decomposition-based semantic auditing framework, the hybrid SCI classification strategy, implementation details, and experimental observations on the FormalRx benchmark.

2. Related works

Autoformalization in Lean 4 The translation of natural language (NL) mathematics into the Lean 4 formal language requires models to master both domain-specific syntax and informal-to-formal reasoning. To address this, the StepFun-Formalizer framework (Wu et al., 2026) leverages knowledge distillation with template-guided reasoning synthesis, enabling models to better understand real-world mathematical contexts and map them precisely into formal Lean expressions. Similarly, the FMC pipeline (Xie et al., 2025) utilizes general-purpose large language models (LLMs) without specialized fine-tuning, employing syntax error feedback loops to translate Olympiad-level problems into Lean 4.

Automated Semantic Alignment Evaluation A critical challenge in autoformalization is ensuring that a syntactically correct formal statement faithfully aligns with the original NL semantics, as compiler checks alone only verify logical validity. The FormalAlign framework (Lu et al., 2025) introduces an automated alignment evaluation method by jointly training a model on sequence generation (using cross-entropy loss) and representational alignment (using contrastive loss), teaching the model to differentiate between semantically aligned and misaligned NL-FL pairs. Another approach to semantic checking bypasses manual verification by measuring the embedding similarity between the original NL text and a back-translated version of the formal statement (Li et al., 2024; Lu et al., 2025). To perform semantic alignment checks without the overhead of intermediate back-translation, the FMC pipeline (Xie et al., 2025) proposes decomposing NL problems into specific data types, conditions, and proof goals, allowing the model to compare this structured format directly against the generated Lean statements. The Decompose, Structure, and Repair (DSR) framework (Liu et al., 2026) translates NL components into Formal Language Operator Trees, providing a hierarchical blueprint that preserves semantic fidelity by explicitly modeling nested operator precedence. Similarly, the Divide and Abstract (DNA) framework (Min et al.) hierarchically decomposes statements into semi-formalized informal clauses, translating each clause using learned mathematical abstractions to maintain precise semantic scope on complex theorems.

3. Method

3.1. Overview

Our system is a multi-stage pipeline for detecting and diagnosing semantic misalignments between informal mathematical statements written in natural language and their corresponding Lean 4 formal statements. Given a pair $(s_{\text{informal}}, s_{\text{formal}})$, the pipeline predicts whether the formalization faithfully preserves the semantics of the informal statement. For misaligned samples, the system additionally generates a minimally corrected Lean 4 statement and classifies the error according to the 28-category Semantic Correctness Issue (SCI) taxonomy introduced in FormalRx (for, 2025).

The pipeline consists of four sequential stages: Informal statement decomposition, Verdict prediction and reasoning, Correction generation and error segmentation, SCI error classification. All language-model inference is performed using `Qwen/Qwen3-235B-A22B` (Yang et al., 2025).

3.2. Stage 1: Informal Statement Decomposition

Motivation. Recent work in autoformalization has shown that decomposing natural-language mathematics into structured semantic components can substantially improve formal translation quality and semantic verification. Frameworks such as FMC (Xie et al., 2025), DSR (Liu et al., 2026), and DNA (Min et al.) demonstrate that explicitly separating mathematical statements into assumptions, goals, or operator-level structures helps preserve semantic fidelity during formalization.

Motivated by these findings, our pipeline adopts a decomposition-first strategy for semantic alignment evaluation. Rather than comparing raw natural-language statements directly against Lean 4 code, we first transform the informal statement into a structured intermediate representation consisting of conditions and conclusions. This representation reduces ambiguity in natural language and exposes the semantic structure of the theorem in a form that is more directly comparable to formal syntax.

Unlike prior decomposition-based approaches that primarily target formal statement generation, our method uses decomposition as a diagnostic representation for semantic auditing. The extracted structure serves as the semantic reference against which the Lean 4 statement is evaluated for incon-

sistencies in quantifiers, hypotheses, operators, constants, domains, and conclusions.

Decomposition prompt. Inspired by DSR (Liu et al., 2026), each informal statement is processed using a dedicated decomposition prompt designed specifically for mathematical semantic extraction. The prompt instructs the model to behave as an "expert mathematician and logic formalizer" and extract two components:

- **Conditions**: premises, assumptions, domain declarations, and hypotheses.
- **Conclusion**: the theorem goal or property to be established.

The prompt imposes several structural constraints to improve consistency and downstream comparability with Lean 4 statements.

Explicit quantifier preservation. A major focus of the prompt is preserving logical quantifier structure. Universal and existential quantifiers are required to remain attached to their predicates rather than being split across conditions and conclusions. For example, statements such as:

$$\textit{For every } n \in \mathbb{N},\ P(n)$$

must be represented as a single quantified formula in the conclusion instead of separating the variable declaration from the predicate.

Similarly, existential statements of the form:

$$\textit{There exists } x \textit{ such that } P(x)$$

must remain intact as a single existentially quantified formula.

This design choice is important because many semantic alignment errors arise from quantifier manipulation, including quantifier strengthening, weakening, or incorrect scope placement.

Atomic condition extraction. The prompt additionally enforces atomic decomposition of premises. Compound assumptions are split into separate logical facts whenever possible. For example:

$$a, b, c > 0$$

is decomposed into three independent conditions:

$$a > 0, \quad b > 0, \quad c > 0.$$

This mirrors the structure commonly used in Lean 4 theorem signatures, where assumptions are typically represented as separate hypotheses.

Implicit type completion. The decomposition stage also explicitly recovers missing domain and type information that may be implicit in natural language statements. If a variable appears free in a formula but its mathematical domain is omitted, the model is instructed to infer and insert the most appropriate type declaration.

For instance, a statement containing $A \subsetneq X$ is decomposed into two separate atomic conditions: the membership declaration $A \subset X$ and the strict-subset property $A \subsetneq X$. Making these implicit assumptions explicit improves compatibility with Lean 4 formalizations, where such declarations are typically required syntactically.

Output format. The decomposition stage outputs a structured representation consisting of:

- a list of atomic conditions,
- a single-line conclusion formula.

The conclusion is always constrained to a single logical expression, even if the original theorem statement contains multiple clauses. This normalization simplifies downstream semantic comparison.

Fallback mechanism. The decomposition stage is implemented as a soft preprocessing component rather than a hard dependency. If decomposition fails due to malformed output or empty extraction results, the pipeline falls back to using the raw informal statement directly in subsequent stages.

3.3. Stage 2: Verdict Prediction

Objective. The second stage determines whether the Lean 4 formal statement is semantically aligned with the informal statement decomposition.

Input representation. The structured decomposition from Stage 1 is concatenated into a semantic context consisting of the extracted conditions and proof goal. This context is provided alongside the original Lean 4 formal statement.

Syntactic correctness and Lean compilation issues are intentionally ignored, since all benchmark statements are guaranteed to compile.

Output format. The model produces two fields:

- a binary verdict: `aligned` or `misaligned`,

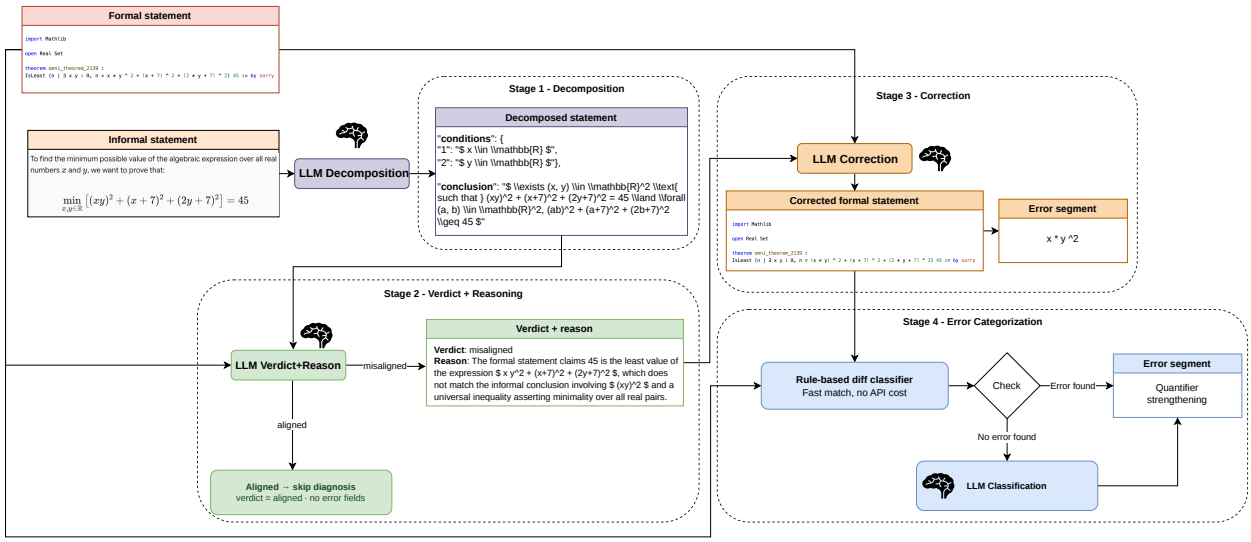

Figure 1. Overview of the proposed semantic alignment verification pipeline. Given an informal mathematical statement and a Lean 4 formal statement, the system first decomposes the informal statement into structured conditions and proof goals. The decomposition is then used for verdict prediction and correction generation. Misaligned samples are further processed by a hybrid SCI classifier consisting of a rule-based token substitution module and a model-based classifier. The final output includes the alignment verdict, SCI error category, error segment, and corrected formal statement.

- a short natural-language reason explaining the decision.

Separating the explanation from the verdict serves two purposes. First, it encourages the model to explicitly identify the semantic discrepancy before committing to a classification decision. Second, the explanation is reused in subsequent stages to guide correction generation and SCI classification.

3.4. Stage 3: Correction Generation

Objective. For samples predicted as misaligned, the pipeline generates a minimally corrected Lean 4 statement that restores semantic alignment with the informal theorem statement.

Conditioned correction generation. Correction generation is performed as a separate model call rather than jointly with verdict prediction. The correction model receives:

- the structured decomposition context
- the original formal statement
- the natural-language reason produced in Stage 2.

Conditioning on the predicted reason is a key design decision. Instead of rediscovering the semantic discrepancy from scratch, the model receives an explicit description of the issue and can focus directly on editing the relevant portion of the theorem statement.

The model is instructed to produce only the corrected theorem signature beginning at the `theorem` keyword, without imports or auxiliary declarations.

Post-processing. In practice, language models occasionally prepend headers. To ensure consistency, a post-processing step removes all content preceding the first occurrence of the `theorem` keyword.

Error segmentation. The erroneous code segment is extracted automatically using token-level diffing between the original and corrected statements. Tokens deleted or replaced in the original statement are collected and concatenated into a single substring. This extracted segment serves both as a localization signal and as an input feature for SCI classification.

3.5. Stage 4: SCI Error Classification

Objective. For misaligned samples with valid corrections, the system assigns an error category from the SCI taxonomy (for, 2025).

The SCI taxonomy (for, 2025) contains 28 semantic error categories organized into three major groups: **Semantic** errors, **Constraint** errors, **Implementation** errors.

Strategy	Verdict F1	Categorization F1	Localization	Correction	Overall
Single-stage	**0.7327**	0.1882	**0.5209**	**0.7671**	0.3179
Single-stage + rule-based error check	**0.7327**	0.2201	**0.5209**	**0.7671**	**0.3397**
Multi-stage	0.4884	0.1880	0.2016	0.5366	0.0732
Multi-stage + rule-based error check	0.5728	**0.2240**	0.2016	0.5366	0.0892

Table 1. Results evaluated on the official AI4Math @ ICML 2026 FormalRx challenge benchmark through the Codabench evaluation server. Overall score denotes the joint per-sample accuracy across verdict prediction, SCI categorization, localization, and correction generation.

3.5.1. Rule-Based Classification

Before invoking the language model classifier, the pipeline first attempts deterministic SCI classification using a rule-based token substitution system. The original formal statement and the corrected statement are tokenized and compared using a token-level diff algorithm based on `difflib.SequenceMatcher`. The resulting removed and added token sets are then matched against a manually curated lookup table of high-confidence substitution patterns.

The rule set targets SCI categories whose semantic signal is both localized and unambiguous, allowing classification through a single symbolic substitution without requiring broader contextual reasoning. These include quantifier swaps ($\forall \leftrightarrow \exists$), logical connective substitutions ($\wedge \leftrightarrow \vee$, $\rightarrow \leftrightarrow \leftrightarrow$), ordering strictness changes ($< \leftrightarrow \leq$), extremum reversals (`IsLeast` ↔ `IsGreatest`), cardinality changes ($\exists \leftrightarrow \exists!$), integration/differentiation confusion (`deriv` ↔ `MeasureTheory.integral`), and common function substitutions such as `Real.sin` ↔ `Real.cos` or `Finset.sum` ↔ `Finset.prod`.

A complete list of deterministic substitution rules is provided in Appendix A. Categories involving structural theorem changes, premise insertion or deletion, numeric constant modifications, or type mismatches are intentionally excluded from the rule-based system due to their higher false-positive risk and are delegated to the model-based classifier instead.

3.5.2. Model-Based Classification

If no rule fires, classification falls back to a language-model classifier. The classifier receives the informal statement, the original formal statement, the extracted error segment, the corrected formal statement, and the reasoning string produced in Stage 2 as input context for SCI category prediction.

The prompt includes the full SCI taxonomy with category descriptions and examples.

Including the Stage 2 reasoning substantially improves disambiguation between closely related categories. For example, explanations such as "the divisibility direction is reversed" strongly indicate an *Incorrect Premise* error rather than a *Missing Premise* error.

3.6. Implementation Details

All experiments use the locally hosted Qwen3-235B model (Yang et al., 2025). Inference is served through the vLLM framework on eight NVIDIA A100 GPUs, enabling efficient batched generation and parallel execution across pipeline stages.

Since the pipeline requires structured outputs at multiple stages, including verdict prediction, correction generation, and SCI categorization, we employ bounded retry logic to handle occasional malformed responses or parsing failures. Whenever the generated output does not conform to the required schema, the corresponding stage is re-invoked up to a fixed maximum number of attempts before returning a failure. This mechanism improves robustness without affecting the underlying model predictions.

4. Results

Table 1 presents the performance of different pipeline variants evaluated on the official AI4Math @ ICML 2026 FormalRx benchmark through the Codabench evaluation server.

The proposed single-stage pipeline achieves the strongest overall performance, obtaining an overall joint accuracy of 0.3397. This configuration also produces the highest Verdict F1 (0.7327), Localization accuracy (0.5209), and Correction accuracy (0.7671), indicating that jointly performing semantic auditing and correction generation within a tightly coupled pipeline is substantially more effective than decomposing the process into multiple sequential reasoning stages.

Adding the rule-based SCI classifier improves Categorization F1 from 0.1882 to 0.2201 without affecting the other evaluation metrics. This demonstrates that deterministic token substitution rules provide a reliable signal for a subset of SCI categories, particularly those involving localized symbolic swaps such as quantifier reversals, logical connective changes, and function confusion errors.

In contrast, the multi-stage variants perform substantially

worse across all metrics. Although decomposition into separate reasoning stages was initially intended to improve interpretability and error isolation, in practice it introduces significant error propagation between stages. Incorrect verdict predictions frequently lead to invalid corrections, which subsequently degrade localization and categorization performance. The rule-based classifier partially mitigates this issue by improving Categorization F1 from 0.1880 to 0.2240, but the overall performance remains significantly below the single-stage pipeline.

These results suggest that semantic alignment evaluation for Lean formalization benefits from tightly integrated reasoning, where verdict prediction and correction generation are solved jointly rather than independently. At the same time, lightweight symbolic heuristics remain valuable as a complementary mechanism for high-confidence SCI categorization.

5. Conclusion

We presented a semantic alignment verification pipeline for Lean 4 autoformalization developed for the FormalRx challenge at AI4Math @ ICML 2026. The proposed system combines structured decomposition of informal mathematical statements, LLM-based semantic auditing, correction generation, and hybrid SCI error classification using both symbolic rules and model-based reasoning. Experimental results on the official benchmark demonstrate that tightly coupled single-stage semantic reasoning substantially outperforms more heavily decomposed multi-stage pipelines. In addition, lightweight rule-based token substitution heuristics provide reliable gains for SCI categorization with minimal computational overhead.

Despite strong performance in verdict prediction and correction generation, categorization and localization remain challenging due to the fine-grained nature of the SCI taxonomy and the structural diversity of Lean formalizations. Future work may explore stronger theorem-aware representations, retrieval-augmented semantic reasoning, and verifier-guided correction mechanisms to improve robustness and interpretability in semantic alignment evaluation.

References

Formalrx: Diagnostic evaluation of autoformalization via structured error taxonomy. In *OpenReview*, 2025. URL https://openreview.net/pdf?id=toNy7vSxa3.

Icml 2026 ai for math workshop & challenge 1 – formalrx, 2026. URL https://www.codabench.org/competitions/15976/.

Li, Z., Wu, Y., Li, Z., Wei, X., Yang, F., Zhang, X., and Ma, X. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. *Advances in Neural Information Processing Systems*, 37:53598–53625, 2024.

Liu, X., Dong, Z., Bai, Y., Li, Y., Liu, Y., and Luo, T. Decompose, structure, and repair: A neuro-symbolic framework for autoformalization via operator trees. *arXiv preprint arXiv:2604.19000*, 2026.

Lu, J., Wan, Y., Huang, Y., Xiong, J., Liu, Z., and Guo, Z. Formalalign: Automated alignment evaluation for autoformalization. In *International Conference on Learning Representations*, volume 2025, pp. 102122–102149, 2025.

Min, M. J., Gao, Y., Sy, W., Li, Z., Si, X., and Bastani, O. Divide and abstract: Autoformalization via decomposition and abstraction learning. In *The Fourteenth International Conference on Learning Representations*.

Wu, Y., Huang, D., Wan, R., Peng, Y., Shang, S., Cao, C., Qi, L., Zhang, R., Zhang, X., Du, Z., et al. Stepfunformalizer: Unlocking the autoformalization potential of llms through knowledge-reasoning fusion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33980–33988, 2026.

Xie, J., Liu, C., Yuan, Y., Li, S., Xiao, Z., and Zhang, M. Fmc: Formalization of natural language mathematical competition problems. *arXiv preprint arXiv:2507.11275*, 2025.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

A. Rule-Based SCI Classification Rules

The rule-based classifier detects a subset of SCI categories using deterministic token substitutions between the original formal statement and the corrected statement. After tokenizing both statements and computing a token-level diff using `difflib.SequenceMatcher`, the system extracts removed and added token sets. If the pair matches one of the predefined swap rules in Table 2, the corresponding SCI category is assigned directly without invoking the language model classifier.

This fast-path classifier is designed only for categories whose semantic signal is an unambiguous symbolic substitution with low false-positive risk. Categories involving structural premise changes, numeric constants, type mismatches, or large-scale theorem rewrites are intentionally excluded and delegated to the model-based classifier.

Removed Token(s)	Added Token(s)	SCI Category
$\forall$	$\exists$	Quantifier Strengthening
$\exists$	$\forall$	Quantifier Weakening
$\wedge$	$\vee$	Logical Connective Misuse
$\vee$	$\wedge$	Logical Connective Misuse
$\rightarrow$	$\leftrightarrow$	Logical Connective Misuse
$\leftrightarrow$	$\rightarrow$	Logical Connective Misuse
$\wedge$	$\rightarrow$	Logical Connective Misuse
$\rightarrow$	$\wedge$	Logical Connective Misuse
$<$	$\leq$	Partial Order Error — Object
$\leq$	$<$	Partial Order Error — Object
$>$	$\geq$	Partial Order Error — Object
$\geq$	$>$	Partial Order Error — Object
`IsLeast`	`IsGreatest`	Extremum Concept Error
`IsGreatest`	`IsLeast`	Extremum Concept Error
`lowerBounds`	`upperBounds`	Extremum Concept Error
`upperBounds`	`lowerBounds`	Extremum Concept Error
`iInf`	`iSup`	Extremum Concept Error
`iSup`	`iInf`	Extremum Concept Error
`sInf`	`sSup`	Extremum Concept Error
`sSup`	`sInf`	Extremum Concept Error
`liminf`	`limsup`	Extremum Concept Error
`limsup`	`liminf`	Extremum Concept Error
$\exists!$	$\exists$	Cardinality Error
$\exists$	$\exists!$	Cardinality Error
`deriv`	`MeasureTheory.integral`	Integration/Differentiation Confusion
`MeasureTheory.integral`	`deriv`	Integration/Differentiation Confusion
`fderiv`	`MeasureTheory.integral`	Integration/Differentiation Confusion
`MeasureTheory.integral`	`fderiv`	Integration/Differentiation Confusion
`deriv`	`intervalIntegral.integral`	Integration/Differentiation Confusion
`intervalIntegral.integral`	`deriv`	Integration/Differentiation Confusion
`Real.sin`	`Real.cos`	Function Confusion
`Real.cos`	`Real.sin`	Function Confusion
`Real.sinh`	`Real.cosh`	Function Confusion
`Real.cosh`	`Real.sinh`	Function Confusion
`Real.log`	`Real.exp`	Function Confusion
`Real.exp`	`Real.log`	Function Confusion
`Finset.sum`	`Finset.prod`	Function Confusion
`Finset.prod`	`Finset.sum`	Function Confusion
`Nat.gcd`	`Nat.lcm`	Function Confusion
`Nat.lcm`	`Nat.gcd`	Function Confusion

Table 2. Deterministic token substitution rules used by the rule-based SCI classifier.