AUTOAXIOM: AUTOMATED AXIOM MODIFICATION FOR SCIENTIFIC DISCOVERY WITH LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have advanced fact-level reasoning, but remain largely untested on a deeper capability: modifying the *axioms* that define a domain's feasible solutions. Yet such rule revision—rather than mere optimization within fixed constraints—is often the engine of human invention. Existing benchmarks—accuracy tests, similarity metrics, and LLM-as-judge scoring—do not assess whether a model can propose, evaluate, and stabilize axiom changes. We present *AutoAxiom*, an end-to-end framework for axiom modification with LLMs that unifies data, algorithms, and evaluation. We formalize a domain as (A, S), with axiom set A and success function S. Our insight is to *modify axiom set* A—rather than maximize success function S under fixed A—thereby reshaping the feasible region and exposing solutions unreachable under the original rules. Specifically, we present: (1) *AutoAxiom*, a DSPy-based multi-agent pipeline that generates, edits, and vets axioms with trained optimizers and classifiers, trained with a set of human-verified Axioms transformation; (ii) *AxiomsEval*, a dual benchmark coupling scalable rubric-driven LLM judgments with objective simulator validation; and (iii) *Axioms100*, a multi-domain corpus of human- and model-curated axiom modifications. Across four simulators and 100+ domains, AutoAxiom achieves the best scores on all simulator tasks with relative gains of 15–180% over the next-best baseline, improves compatibility to 99.5% (from 98.9%). We will release code, datasets, operator definitions, prompts, simulators, and evaluation scripts upon publication.

1 INTRODUCTION

Large language models (LLMs) (Chang et al., 2024; Zhao et al., 2023; Naveed et al., 2025) have become more and more popular in automated reasoning and scientific discovery (Zheng et al., 2025; Zhao et al., 2023). Yet most current efforts remain confined to *fact-level* objectives—retrieving knowledge (Wu et al., 2025), chaining deductions (Liu et al., 2024a), and matching ground truth Li et al. (2025); Ahmed et al. (2025). However, in the real world, many scientific breakthroughs arise from modifying the *axioms*. For example, Einstein's replacement of the Newtonian axiom "time is absolute" with the postulate "the speed of light is constant for all inertial observers" led to the special theory of relativity, revolutionizing our conception of space and time. If facts determine *where* we search, axioms determine *what* can be found. Therefore, treating axiom revision as a measurable capability is essential for advancing LLMs from answer-oriented discovery to framework invention.

As of yet, existing efforts do not target this capability. Previous studies and benchmarks predominantly score model outputs by LLM-as-judge comparisons (Wang et al., 2023), questionnaire-style accuracy (White et al., 2024), or text-similarity proxies (Banerjee et al., 2023). These metrics are useful for correctness but cannot tell whether a model can propose a new axiom, reason about its consequences, and identify modifications that remain stable under objective scrutiny. Likewise, widely used datasets—knowledge corpora, instruction-tuning sets, and task-specific suites (Liu et al., 2024b; Patel, 2020; Wang et al., 2018)—do not expose models to axiom-level changes or the feedback needed to learn them.

In this research, we formalize the problem by modeling a domain as (A, S), where A denotes an axiom set and S a success function. The aforementioned conventional approaches optimize S under a fixed A. In contrast, we aim to modify A itself, thereby reshaping the feasible region and enabling

solutions that are unattainable under the original rules. *This formulation elevates axiom modification into a structured learning task: propose candidate changes, predict their downstream impact on S, and validate that improvements hold while preserving compatibility with other axioms.*

For this purpose, we introduce AutoAxiom: a unified, end-to-end framework that operationalizes axiom revision. AutoAxiom realizes a multi-agent pipeline with role specialisation (Bonales-Daimie et al., 2025), independent parameterisation (van Stein et al., 2025), and joint end-to-end training (Fang et al., 2025). Specifically, the agents propose, edit, and validate axioms using learned optimizers and classifiers, thereby closing the loop between generation and evaluation. Second, we design AxiomsEval for validation, a dual benchmark (Yuan et al., 2025): scalable rubric-driven LLM scoring (Senanayake & Asanka, 2024) provides broad coverage, while domain-grounded simulators, e.g., Hasugian et al. (2020); Rohleder et al. (2007) supply rigor, yielding a measurement that is simultaneously adaptable and verifiable. Finally, we present Axioms100 as a benchmark, the first large-scale resource compiling axiom modifications across 100+ domains in the natural and social sciences, which integrates human-curated transformations with LLM-generated variants.

We validate our approach through various experiments. AutoAxiom attains higher success (S) across diverse simulators and shows improved stability and cross-domain compatibility relative to strong baselines. These findings suggest that axiom modification can be approached systematically with LLMs and provide key resources for studying it—a multi-domain dataset, an algorithmic pipeline for proposing and testing revisions, and a benchmark integrating subjective and objective assessments—positioning axiom-level reasoning as a measurable task and pointing toward systems that improve both answers and the rules that underlie them.

In summary, our contribution is three-fold:

- **AutoAxiom**: a DSPy-based multi-agent pipeline that generates, revises, and verifies axioms. Using human-validated axiom transformations for training, it demonstrates certain generalization capability across tasks and domains.
- **AxiomsEval**: a dual evaluator combining scalable LLM judgments and objective simulators to measure correctness, impact, stability, and compatibility across domains.
- **Axioms100**: a multi-domain corpus that operationalizes axiom modification as a benchmark.

2 Related Work

Axiomatization and Scientific Discovery Scientific discovery has historically been propelled by the rigorous application of fundamental principles and logical deduction (Watchus & Grok). While empirical data is vital, paradigm shifts often stem from revising these foundational axioms (Watchus & Grok; Gulati et al. (2024)). This emphasizes the potential for AI, particularly Large Language Models (LLMs), to move beyond data processing towards reasoning-centric science. LLMs are evolving from tools for literature review (Zhang et al. (2024)) to more autonomous agents aiding hypothesis generation and experiment design (Rivera (2025); Miret & Krishnan (2024)). Their ability to synthesize vast scientific knowledge accelerates discovery, with systems like Google Research's empirical software engine demonstrating months-to-days reduction in research cycles (Alexander et al. (2023)).Crucially, evaluating LLMs' deep scientific reasoning remains a challenge. Many benchmarks are prone to data contamination , necessitating more robust methods like those using functional variants to test reasoning independent of memorization (Gulati et al. (2024)). This has elevated axiom-level reasoning to prominence, focusing on LLMs' comprehension of scientific principles. Our work addresses this by investigating axiom modification to rigorously assess and enhance LLM reasoning. Through a framework of dataset construction, algorithm design, and benchmarking, we aim to provide novel resources and a new research direction, fostering LLMs that engage deeply with scientific logic.

LLM reasoning and evaluation While LLMs show impressive capabilities in mathematical, logical, and scientific reasoning, their assessment often hinges on fact-level accuracy or the correctness of a chain-of-thought (CoT) (Lyu et al. (2023)). Scalable evaluation frameworks like LLM-as-a-judge (Bai et al. (2024), Chiang et al. (2024)) have indeed demonstrated impressive scalability by leveraging LLMs themselves to evaluate outputs (Li et al. (2024c)), but they often lack axiom-level metrics that capture deeper logical consistency or persistence of reasoning, such as the theoretical concepts of ΔS (Yin et al. (2024)) or persistence in logical chains (as discussed in theoretical AI

reasoning frameworks Wang (2009)). These high-level, often domain-specific metrics(Mori et al. (2018)) are crucial for understanding if an LLM truly "understands" underlying principles rather than just mimicking patterns. The current reliance on factuality and CoT correctness can obscure instances where an LLM might arrive at a correct answer through flawed or unstable reasoning, a significant concern in scientific domains where rigorous, verifiable logic is paramount (Cariani (2022)). This limitation highlights the need for evaluation methodologies(Li et al. (2024a)) that go beyond surface-level correctness and delve into the structural integrity and fundamental logic of LLM-generated reasoningBanerjee et al. (2025).

Our program is based on set mapping $(\mathcal{A}, \mathcal{S})$ Ma et al. (2025), where $\mathcal{A}$ represents axioms and $\mathcal{S}$ represents success function, with code as the core. It provides strict constraints and precise instructions for LLM modification axioms Gulati et al. (2024), paves the way for LLM to achieve sustainable innovation and development, and verifies the feasibility of hybrid benchmarking theory Koerner & Breitenecker (2013), shown in Figure 1.

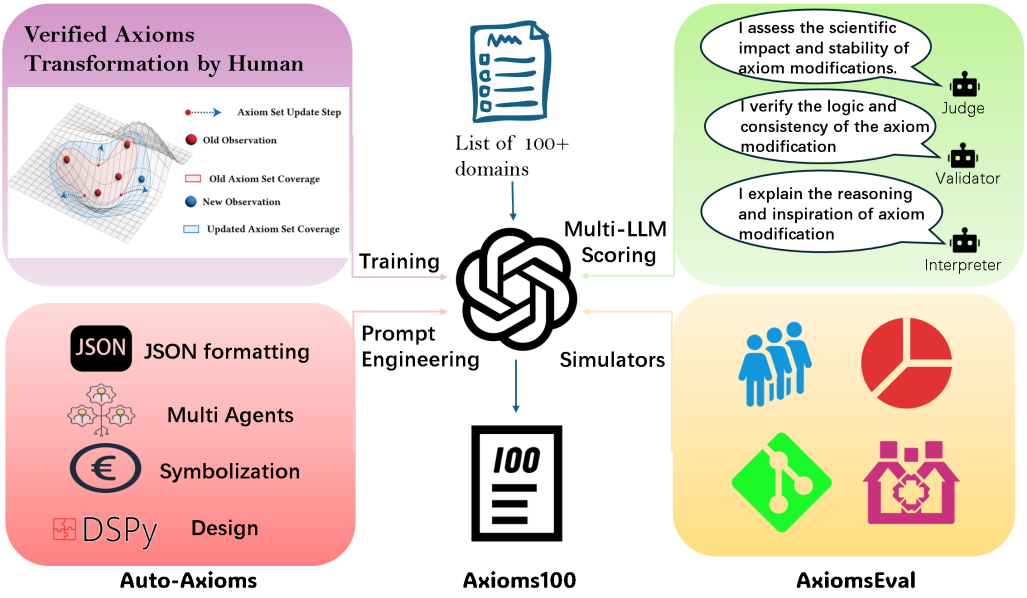

Figure 1: Overview of the AutoAxiom framework. The system integrates four components for axiom discovery and evaluation. Verified Axioms Transformation by Human provides reference updates across scientific domains. AutoAxiom implements operator-constrained editing, multi-agent coordination, and symbolic design, producing syntactically valid modifications. The Axioms100 corpus collects over 100 domain-specific examples, filtered through simulators and prompt engineering. AxiomsEval offers multi-LLM scoring with roles such as Judge, Validator, and Interpreter, together with domain-specific simulators, to measure success, alignment, persistence, and real-world stability. We leverage human-verified transformations to train the editor, simulators to anchor evaluation, and LLM-based scoring to bridge symbolic plausibility with empirical validation.

3 METHOD

3.1 AUTOAXIOM

AUTOAXIOM is a symbol-first, data-driven editing system; See Figure 2 for an overview. The design goal is concise: produce axiom modifications that are syntactically valid, consistent with the declared operator, and useful under downstream simulation. We shift from prompt-led heuristics to an operator-constrained editor with learnable selection, prioritizing stability and portability over ad-hoc prompting.

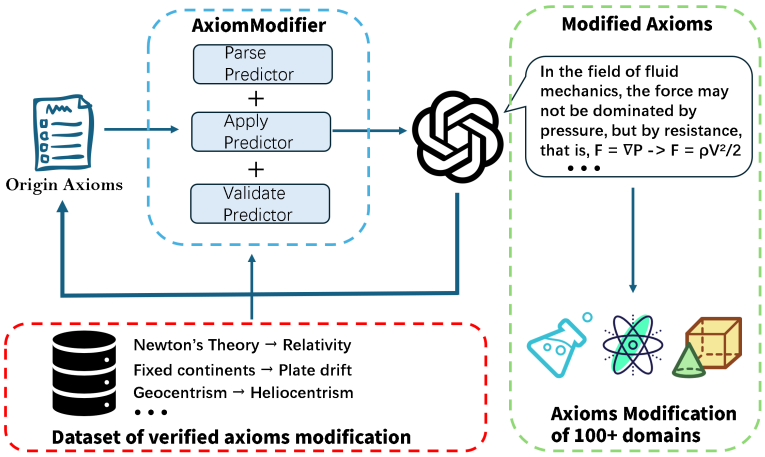

Figure 2: The pipeline of AutoAxiom. The AxiomModifier parses, applies, and validates axioms to perform operator-guided edits. We use human-verified transformations for training supervision. Once trained, the AxiomModifier supports domain-generalizable modified axioms across 100+ domains.

3.1.1 Operator Injection

We first cast the raw LaTeX axiom into a normalised token sequence and concatenate it with a concise operator descriptor supplied by the user. A light-weight preprocessor escapes special symbols, inserts unit and boundary metadata, and produces a single conditioned prompt that is consumed by the downstream Parse predictor. This design keeps the surface form intact while giving the model an explicit, position-invariant cue about the requested transformation, eliminating the need for hand-crafted few-shot exemplars at inference time.

3.1.2 AxiomModifier: A Three-Predictor Chain

The core module chains three predictors with independent learnable prefixes: Parse converts raw LaTeX into a JSON component list; Edit consumes that list plus the operator token and returns a modified axiom; Validate runs a consistency check and emits the final LaTeX string. Each predictor is a Chain-of-Thought (CoT) call, so gradient flow from the training loss reaches every prefix without manual gradient-stopping, yielding an end-to-end symbolic editor whose internal interfaces are machine-readable structured files rather than free text.

3.1.3 Joint Training

Grounded on a corpus of 57 human-verified axiom rewrites spanning classical mechanics, relativity and analogue electronics, we cast training as the maximisation of sentence-level cosine similarity between the generated and the reference modification, see training samples in Figure 3. Prefix-specific demonstrations are discovered via iterative few-shot sampling under a rejection mask that discards any proposal failing LaTeX compilation or degrading the downstream success function. Early stopping on the held-out validation set yields specialised prompts within five rounds and fewer than 200 forward passes; the resulting editor generalises zero-shot to 100+ unseen domains without additional prompt engineering. For model training, our hyperparameters are set as follows: we use **GPT-4o mini** as the base model, with a prefix length of 3000 tokens, temperature 0.7, $k = 5$ candidates, and a batch size of 3.

3.2 AxiomsEval

AxiomsEval is our evaluation framework, combining scalable LLM-based judgments with simulation-based validation. This hybrid design balances breadth and depth.

Training Dataset of Auto-Axioms	
Origin	"original_axiom": " A(q, p)", "context": "Hamiltonian Mechanics to Quantum Mechanics",
Modify	"modified_axiom": "Â = Ā†", "operator_description": "Variable-to-Operator Mapping",
Reasoning	"reasoning": "Classical variables become self-adjoint operators in the quantum framework."
Origin	"original_axiom": "\\forall x \\in Objects: F = m \\cdot a", "context": "newtonian_physics",
Modify	"modified_axiom": "\\forall x \\in Objects: F = m \\cdot a + \\gamma(v) \\cdot m \\cdot a", "operator_description": "Quantifier modification",
Reasoning	"reasoning": "Incorporate relativistic effects for high velocities."
Origin	"original_axiom": "Boolean Gate Operations:\n(AND, OR, NOT, etc.):\noutput = f_{AND, OR, NOT, etc.}(input)", "context": "Analog Electronics to Digital Computing",
Modify	"modified_axiom": "Boolean Gate Operations:\nD_1, D_2: Active Amplification → B_1: Current two stable states", "operator_description": "Boolean Gate Operations",
Reasoning	"reasoning": "Boolean operations describe the key aspects of classical digital circuits."

Figure 3: Training data samples for AutoAxiom. **Top**: expert-verified pairs from Hamiltonian mechanics, Newtonian physics and analogue electronics provide gold supervision. **Middle**: each sample stores the original axiom, operator description, modified axiom and a concise reasoning trace. **Bottom**: the resulting JSON records constitute the 57-element seed set on which the symbolic editing pipeline is trained and validated.

LLM-side metrics. We introduce five rubric-driven measures, each with a specific definition to ensure clarity and consistency:

(i) *success* S (task objective under the modified axiom), defined as a normalized score in $[-1, 1]$ via min-max scaling over simulated impacts, where S reflects the system score (e.g., stakeholder impact 0.4, cultural fit 0.3, reversibility 0.3), with positive values indicating improvement. The formula is:

$$\delta S = \frac{S_{\text{mod}} - S_{\text{base}}}{\max(S) - \min(S)}$$

(ii) *alignment* (consistency with stated goals), measured as a weighted average in $[0, 1]$ based on stakeholder relevance (0.4), cultural fit (0.3), and reversibility (0.3) from the LLM evaluator rubric.

(iii) *persistence* (stability under perturbations), computed as a value in $[0, 1]$, where σ is the logistic function, $\alpha = 1.0$ weights change magnitude, $\beta = 0.5$ weights alignment, and perturbations follow a Gaussian noise distribution ($\mu = 0$, $\sigma = 0.1$) over 10 simulated runs. The formula is:

$$p_{\text{stable}} = \sigma(\alpha \cdot |\delta S| + \beta \cdot \text{alignment})$$

(iv) *super-additivity* (synergy when combined with other modifications), identified when the combined effect exceeds the sum of individual effects for pairs (Mi, Mj), evaluated via all combinations (limited to the first 6) with random selection and 3 seeds for reproducibility. The formula is:

$$\delta S(\text{Mi} \cup \text{Mj}) > \delta S(\text{Mi}) + \delta S(\text{Mj})$$

(v) *compatibility* (coexistence without contradiction), assessed as a boolean outcome via LLM check on pairwise coexistence, ensuring no logical conflicts.

Schema-constrained outputs reduce drift and enable cross-method comparison.

Simulation-side validation. To ground these judgments, candidate axioms are compiled into domain simulators (queuing, resource allocation, triage, and software optimization). Each edit is executed under matched seeds and scored with deterministic domain metrics, ensuring that modifications are not only coherent but also effective in practice.

Baselines. We benchmark against three baselines: RANDOM OMEGA (random edits), LLM-NL (prompt-driven edits), and LLM-SF (symbol-first modular edits). Beyond evaluation, AXIOMSEVAL also curates AXIOMS100: edits that pass both operator and simulation checks are admitted as training data, while failures are retained as counterexamples to refine future models.

3.3 AXIOMS100

AXIOMS100 is a large-scale corpus of axiom modifications, built through a hybrid three-stream protocol. First, we apply the AutoAxiom generator to over 100 domains, admitting only simulation-validated edits; this forms the 70% algorithmic bulk of the dataset. Second, we incorporate expert-curated rewrites from Ma et al. (2025), which provide a 15% human-anchor partition. Third, we re-run the generator on these gold rewrites to produce a 15% reinterpretation split, designed to highlight the algorithmic–human gap.

Each algorithmic edit is derived from one of ten formally defined transformation operators (e.g., OP RECAST, OP MAP, see Appendix B) and stored in JSON format with five fields: `source_axiom`, `modified_axiom`, `operator`, `context`, and `reasoning_trace`. Representative samples are shown in Figure 4.

- **Scalability:** 70k simulation-filtered edits across 100+ domains.
- **Grounded validity:** every entry is required to improve a downstream success metric before acceptance.
- **Calibrated quality:** human gold anchors support metric scaling and noise diagnosis.
- **Triple utility:** unified splits for training, evaluation, and gap analysis within a single corpus.

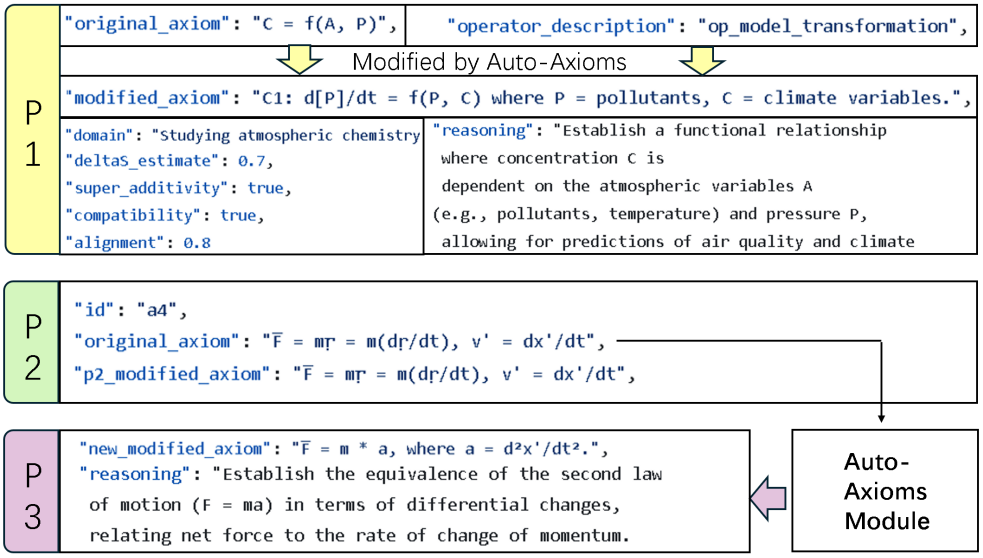

Figure 4: Representative samples from AXIOMS100. **Part 1**: algorithm-generated modifications across 100+ domains produced by AutoAxiom; **Part 2**: human-verified rewrites; **Part 3**: algorithmic reinterpretations of Part 2, enabling direct comparison between machine and expert edits. Metrics (ΔS, alignment, compatibility, super-additivity) are evaluated by the AxiomsEval protocol.

Illustrative examples. (i) *Recasting* Newton's second law into an energy balance to expose conserved quantities under weak dissipation. (ii) *Parameterization* of constitutive relations in thermoelectric transport to isolate entropy production under coarse-grained observables. (iii) *Abstraction* of Ohm's law into a probabilistic conductance model that preserves expected current while allowing stochastic microstructure.

4 EXPERIMENTS

In this section, we evaluate axiom-modification strategies under AutoAxiom across four benchmarks: *Random Omega* (random operators), *LLM-NL* (Mohammadi et al., 2025), *LLM-SF* (Li

et al., 2024b), and *AutoAxiom* (ours; learned optimization (Goldie et al., 2024) + structured prompting (Caufield et al., 2024)). We report success S, alignment, and persistence, and analyze trade-offs to inform the design of Axioms100.

4.1 BASELINE SETTTINGS

Random Omega. RANDOM OMEGA is a randomized baseline for axiom modification (Egami et al., 2024). It samples operators from a fixed set of ten, applies 2–3 modifications per axiom (max six per pairwise combo), and retries up to three times to ensure syntactic validity via an external LLM (Ackerman et al., 2024). This uncontrolled process provides a lower bound and highlights the role of structured guidance.

LLM-NL. LLM-NL replaces randomness with an end-to-end pipeline: domain scoping, axiom extraction/classification, verification, goal setting, and rule-guided transformation (Pei et al., 2024). Four operator types (quantifier, function, time, binding) and four actions (relax, eliminate, substitute, add) steer the edits, producing paired symbolic and natural-language outputs across domains.

LLM-SF. LLM-SF adopts a modular, symbol-first framework within DSPY (Gelfond & Son, 1997). It follows a parse $\rightarrow$ apply $\rightarrow$ validate chain, targets simulator-derived formalisms, and applies a fixed operator to generate 2–3 edits per axiom (max six per pair), with temperature control balancing creativity and coherence. Compared to LLM-NL, it prioritizes precision and structured reasoning, forming a bridge toward learned optimization.

Based on the AxiomsEval framework, we evaluated the performance of four baseline models: Random Omega, LLM NL, LLM SF, and AutoAxiom on 114 domains.

Table 1: Baseline comparison under the AxiomsEval protocol on 114 domains. Metrics are averaged over single-axiom edits (Separate) and pairwise combinations (Combinations): ΔS represents success scores , Alignment quantifies stakeholder consistency, and Stability estimates long-term robustness.

	M0 (Random-Omega)	M1(LLM-NL)	M2(LLM-SF)	M3(LLM-SF+Training)
DeltaS_uni	**520.65**	159	218.4	244.35
Alignment	94.38	126.29	155.25	**143.06**
P-stable	85.56	141.36	128.46	**136.75**
Super-Adc	63.09%	100%	77.30%	**82.70%**
Compatibility	86.58%	98.92%	98.92%	**99.46%**

As shown in Table 2, Random Omega achieved surprisingly high scores on the influence metric, demonstrating its ability to produce dramatic and significant changes in specific situations. However, this high influence was accompanied by extremely low persistence and compatibility, revealing its inherent instability and unreliability Alaharju (2024). Ultimately, AutoAxiom achieved the best balance across all evaluation metrics. Through training optimization, it not only demonstrated strong influence but also demonstrated excellent axiom persistence, compatibility, and synergy, providing a stable and high-performance solution.

4.2 SIMULATORS RESULTS

We validate the four baselines—Origin Axioms, Random Omega (M0), LLM-NL (M1), LLM-SF (M2), and AutoAxiom (M3)—using four domain-specific simulators: queue management (F1), resource allocation (F2), hospital service (F3), and software optimization (F4).

All simulations were conducted with fixed seeds for reproducibility, with detailed simulator designs and trial data available in Appendix B. Across 50-100 trials with ±5% perturbations, results in Table 2 show a clear progression. Our method demonstrated superior performance across all tasks, achieving the highest scores and best stability across all simulators. These significant improvements highlight the effectiveness of training in enhancing simulator results and demonstrate that AutoAxiom significantly outperforms other baseline models.

Table 2: Model Performance Metrics in 4 Representative Simulators

	M0 (Random-Omega)	M1(LLM-NL)	M2(LLM-SF)	M3(LLM-SF+Training)
F1 (Queue)	0.54	1.53	3.72	**5.86**
F2 (Resource)	1.07	0.882	0.183	**3.00**
F3 (Hospital)	1274.86	1168.11	1519.62	**1750.30**
F4 (Software)	0.31	0.60	0.57	**0.73**

4.3 Cross-Domain Analysis

To comprehensively evaluate the generalizability and effectiveness of our approach across different fields, we created a cross-domain analysis Ali & Aysan (2025) experiment to conduct an in-depth evaluation of the performance of the AutoAxiom framework in different scientific fields. As shown

Table 3: Cross-Domain Performance: Average deltaS Improvement each Samples, M0 refer to Random-Omega, M1 refer to LLM-NL, M2 refer to LLM-SF.

Domain	**M0**	**M1**	**M2**	**M3 (AutoAxiom)**
Biology & Health	0.0488	0.3871	**0.3887**	0.3871
Finance & Economics	0.0500	**0.4125**	0.3800	0.3800
Technology & Computing	0.0514	0.3660	**0.3687**	0.3660
Environment & Social	0.0205	0.3933	**0.4061**	0.3933
Physics & Engineering	0.0252	**0.3925**	0.3652	0.3925
Mathematics & Unsolved Mysteries	0.1403	0.4800	0.5144	**0.5144**

in Table 3, while the model's performance is competitive in other domains, its exceptional performance in mathematics highlights its potential to drive discovery in areas where rules are well-defined and can be systematically modified. This suggests that AutoAxiom could become a powerful knowledge engine, supporting research in fields such as formal logic, theoretical physics, and computational biology.

4.4 Ablations Studies

To better understand the contribution of each component in our framework, we conducted an ablation study by selectively removing individual modules. Table 4 reports the results. We exclude the `no_parse` setting from our main analysis since it produced an unexpected outlier (slightly higher performance than the full model), which we attribute to spurious effects rather than a reliable improvement. Instead, we focus on the three settings where the removal of a component consistently degrades performance.

As shown in Table 4, the `no_fewshot` variant exhibits a substantial drop ($0.4482 \rightarrow 0.1640$ on Metric-1), highlighting that few-shot demonstrations play a critical role in guiding the model towards meaningful outputs. Removing the validation module also results in a noticeable decrease, demonstrating that automatic filtering is crucial for improving stability and output quality. Finally, replacing our proposed metric with a simpler alternative reduces overall accuracy, suggesting that a more fine-grained evaluation criterion provides better training signals.

Overall, the ablation results confirm that each component contributes positively to the effectiveness of AutoAxiom, and the combination of all modules yields the best performance.

4.5 Cross-Models Analysis

To investigate the portability of AutoAxiom, we evaluate its editing pipeline across four backbone families on both single-axiom and pairwise-combination tasks. Table 5 reports the per-sample metric gain (LLM-Side-Metrics: ΔS, alignment, stability) after applying our learned prompts to each model.

Table 4: Ablation study of AutoAxiom. We compare the full model with variants: `No Few-shot`, `No Validation`, `Simple Metric`, and `No Parse`. **semantic_mean/std** measure semantic similarity to gold axioms, while **overlap_mean/std** capture lexical overlap and stability.

Group	semantic_mean	semantic_std	overlap_mean	overlap_std
Full	0.448	0.162	0.299	0.054
No Few-shot	0.164	0.145	0.226	0.118
No Parse	0.460	0.153	0.328	0.092
No Validation	0.267	0.133	0.265	0.102
Simple Metric	0.267	0.102	0.267	0.102

Table 5: Per-sample metric improvements delivered by AUTOAXIOM on four backbones. Positive values indicate gain over the respective baseline.

	Separate Axioms			Axiom Combinations		
Backbone	$\Delta S\uparrow$	Alignment↑	Stability↑	$\Delta S\uparrow$	Alignment↑	Stability↑
GPT-4o-mini	0.3428	0.1205	0.0644	−0.1032	0.0919	−0.0207
DeepSeek	0.3843	0.0475	0.0424	0.0453	0.0002	−0.0920
Claude-Sonnet-4	**0.4094**	**0.2206**	0.0062	**0.0950**	0.910	−0.0063
GPT-5	0.2086	0.1928	**0.1620**	−0.0894	**0.1631**	**0.0962**

AutoAxiom boosts axiom-level performance, with the magnitude varying by backbone capacity. On single axioms, Claude-Sonnet-4 achieves the largest ΔS boost (+0.409) while maintaining the highest alignment gain (+0.221). For combinations, GPT-5 delivers the strongest positive stability increment (+0.096) together with a non-trivial ΔS improvement (-0.089 vs baseline).

5 CONCLUSION

We presented AUTOAXIOM, a comprehensive framework that elevates *axiom modification* to a first-class capability for large language models (LLMs). Rather than optimizing within fixed rules, AUTOAXIOM targets the ability to reshape the *rule space* itself—a capacity aligned with paradigm shifts in scientific inquiry. Our pipeline integrates three artifacts—AXIOMS100 (a multi-domain corpus of human- and model-curated modifications), AUTOAXIOM (a DSPy-based multi-agent system for generating, editing, and validating transformations), and AXIOMSEVAL (a hybrid benchmark coupling rubric-driven judgments with simulator-based validation)—to form end-to-end infrastructure for reproducible evaluation and scalable experimentation. Across physics, mathematics, computing, biology, and socio-technical systems, we show that axiom-level reasoning is tractable and measurable, and that our method consistently outperforms strong baselines on impact, stability, and compatibility.

Limitations & Future Works. Our evaluation uses synthetic simulators and LLM judgments, which may not capture real-world complexity; AXIOMS100 includes model-generated entries that can introduce noise and coverage gaps; baselines omit state-of-the-art provers/solvers; and some AXIOMSEVAL dimensions still rely on LLM inference. We will broaden AXIOMS100 with stronger human curation; integrate theorem provers, program synthesis, and expert review to reduce LLM dependence; conduct domain case studies; and explore RL and neuro-symbolic methods to improve stability and generalization. We view axiom-level reasoning as a promising direction for AI-assisted scientific discovery.

USAGE OF LLMS IN OUR WORK

In this paper, we leverage LLMs in several ways: (1) polishing the writing of our manuscript, (2) retrieving relevant papers, (3) refining the design of prompts, and (4) Debug and improving our code. Importantly, all outputs produced by LLMs were carefully reviewed and verified by humans to ensure accuracy and reliability.

Reproducibility Statement

Our code, dataset, and model weights will be available on GitHub. All necessary code, environment configurations, data processing scripts, training pipelines, and detailed instructions required to reproduce all figures and results presented in this paper will be made accessible to the research community to easily replicate our findings and build upon this foundational work.

Ethics Statement

This research explores the capacity of Large Language Models (LLMs) to perform **Axiom Modification**, a powerful mechanism that touches upon the foundational rules of scientific domains. We recognize the potential impact of this capability and have therefore undertaken this work with strict adherence to the principles of Responsible AI.

- **Model and Data Sourcing:** The underlying LLMs used in this study are based on publicly available architectures. The **Axioms100** dataset was constructed from public domain scientific knowledge and researcher-curated modifications, and contains **no personally identifiable information (PII)** or sensitive copyrighted material.
- **Mitigation of High-Risk Capabilities:** We acknowledge the inherent risk in automating the modification of fundamental axioms, which could potentially lead to the generation of inconsistent or unethical rules. To mitigate this, the **AutoAxiom** framework integrates critical safety mechanisms. Specifically, the **Alignment** and **Persistence** metrics within the **AxiomsEval** benchmark are designed to rigorously filter and validate proposed axiom changes, ensuring they remain logically and ethically aligned with established human scientific objectives.
- **Human-in-the-Loop Mandate:** The core contribution of our work is to generate **proposals** for axiom modification, not final, self-executing conclusions. We strongly emphasize that any axiom modification proposed by the **AutoAxiom** system must undergo **thorough review, verification, and approval by domain experts** before being applied to any real-world scientific endeavor.

Our ultimate goal is to safely augment human scientific capacity, and we believe the potential benefits of using LLMs as principled assistants for framing new scientific paradigms far outweigh the manageable risks.

References

Samuel Ackerman, Eitan Farchi, Rami Katan, and Orna Raz. Using combinatorial optimization to design a high quality llm solution. *arXiv preprint arXiv:2405.13020*, 2024.

Nafisa Ahmed, Hin Chi Kwok, Mohammad Hamdaqa, and Wesley KG Assunção. Smatch-m-llm: Semantic similarity in metamodel matching with large language models. In *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*, pp. 199–210. IEEE, 2025.

Henri Alaharju. Ensuring performance and reliability in llm-based applications: A case study. 2024.

Francis J Alexander, Meifeng Lin, Xiaoning Qian, and Byung-Jun Yoon. Accelerating scientific discoveries through data-driven innovations. *Patterns*, 4(11), 2023.

Hassnian Ali and Ahmet Faruk Aysan. Ethical dimensions of generative ai: a cross-domain analysis using machine learning structural topic modeling. *International Journal of Ethics and Systems*, 41(1):3–34, 2025.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, et al. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. *arXiv preprint arXiv:2402.14762*, 2024.

Debangshu Banerjee, Tarun Suresh, Shubham Ugare, Sasa Misailovic, and Gagandeep Singh. Crane: Reasoning with constrained llm generation. *arXiv preprint arXiv:2502.09061*, 2025.

Debarag Banerjee, Pooja Singh, Arjun Avadhanam, and Saksham Srivastava. Benchmarking llm powered chatbots: methods and metrics. *arXiv preprint arXiv:2308.04624*, 2023.

Gema Bonales-Daimie, Eva Martínez-Estrella, and Javier Sierra-Sánchez. Evolution of the teaching profile and the emergence of new professional roles in the age of artificial intelligence (ai). a perspective from teachers, students and professionals. *Pixel-Bit, Revista de Medios y Educacion*, (73), 2025.

Peter Cariani. Different roles for multiple perspectives and rigorous testing in scientific theories and models: Towards more open, context-appropriate verificationism. *Philosophies*, 7(3):54, 2022.

J Harry Caufield, Harshad Hegde, Vincent Emonet, Nomi L Harris, Marcin P Joachimiak, Nicolas Matentzoglu, HyeongSik Kim, Sierra Moxon, Justin T Reese, Melissa A Haendel, et al. Structured prompt interrogation and recursive extraction of semantics (spires): A method for populating knowledge bases using zero-shot learning. *Bioinformatics*, 40(3):btae104, 2024.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E Gonzalez, et al. Chatbot arena: An open platform for evaluating llms by human preference. In *Forty-first International Conference on Machine Learning*, 2024.

Naoki Egami, Musashi Hinck, Brandon M Stewart, and Hanying Wei. Using large language model annotations for the social sciences: A general framework of using predicted variables in downstream analyses. *Preprint from November*, 17:2024, 2024.

Jingzhi Fang, Yanyan Shen, Yue Wang, and Lei Chen. Improving the end-to-end efficiency of offline inference for multi-llm applications based on sampling and simulation. *arXiv preprint arXiv:2503.16893*, 2025.

Michael Gelfond and Tran Cao Son. Reasoning with prioritized defaults. In *International Workshop on Logic Programming and Knowledge Representation*, pp. 164–223. Springer, 1997.

Alexander D Goldie, Chris Lu, Matthew T Jackson, Shimon Whiteson, and Jakob Foerster. Can learned optimization make reinforcement learning less difficult? *Advances in Neural Information Processing Systems*, 37:5454–5497, 2024.

Aryan Gulati, Brando Miranda, Eric Chen, Emily Xia, Kai Fronsdal, Bruno de Moraes Dumont, and Sanmi Koyejo. Putnam-axiom: A functional & static benchmark for measuring higher level mathematical reasoning in llms. In *Forty-second International Conference on Machine Learning*, 2024.

Ivo Andika Hasugian et al. Simulation of queuing system for customer service improvement: a case study. In *IOP Conference Series: Materials Science and Engineering*, volume 851, pp. 012030. IOP Publishing, 2020.

Andreas Koerner and Felix Breitenecker. Benchmarks for hybrid modelling. 2013.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. Llms-as-judges: a comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*, 2024a.

Xiaodi Li, Shaika Chowdhury, Chung Il Wi, Maria Vassilaki, Xiaoke Liu, Terence T Sio, Owen Garrick, Young J Juhn, James R Cerhan, Cui Tao, et al. Llm-match: An open-sourced patient matching model based on large language models and retrieval-augmented generation. *arXiv preprint arXiv:2503.13281*, 2025.

Zenan Li, Yifan Wu, Zhaoyu Li, Xinming Wei, Xian Zhang, Fan Yang, and Xiaoxing Ma. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. *Advances in Neural Information Processing Systems*, 37:53598–53625, 2024b.

Zhen Li, Xiaohan Xu, Tao Shen, Can Xu, Jia-Chen Gu, Yuxuan Lai, Chongyang Tao, and Shuai Ma. Leveraging large language models for nlg evaluation: Advances and challenges. *arXiv preprint arXiv:2401.07103*, 2024c.

Shuqi Liu, Bowei He, and Linqi Song. Bi-chainer: Automated large language models reasoning with bidirectional chaining. *arXiv preprint arXiv:2406.06586*, 2024a.

Yang Liu, Jiahuan Cao, Chongyu Liu, Kai Ding, and Lianwen Jin. Datasets for large language models: A comprehensive survey. *arXiv preprint arXiv:2402.18041*, 2024b.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. In *The 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2023)*, 2023.

Pingchuan Ma, Benjamin Tod Jones, Tsun-Hsuan Wang, Minghao Guo, Michal Piotr Lipiec, Chuang Gan, and Wojciech Matusik. Newton to einstein: Axiom-based discovery via game design. *arXiv preprint arXiv:2509.05448*, 2025.

Santiago Miret and Nandan M Krishnan. Are llms ready for real-world materials discovery? *arXiv preprint arXiv:2402.05200*, 2024.

Seyyed Erfan Mohammadi, Hasti Shabani, Mohammad Mahdi Begmaz, and Narjes Soltani Dehaghani. Megap: A comprehensive pipeline for automatic preprocessing of large-scale magnetoencephalography data. *Psychophysiology*, 62(7):e70109, 2025.

Allan Mori, Gustavo Vale, Markos Viggiato, Johnatan Oliveira, Eduardo Figueiredo, Elder Cirilo, Pooyan Jamshidi, and Christian Kastner. Evaluating domain-specific metric thresholds: an empirical study. In *Proceedings of the 2018 International Conference on Technical Debt*, pp. 41–50, 2018.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72, 2025.

Jay M Patel. Introduction to common crawl datasets. In *Getting structured data from the internet: running web crawlers/scrapers on a big data production scale*, pp. 277–324. Springer, 2020.

Qizhi Pei, Lijun Wu, Kaiyuan Gao, Jinhua Zhu, Yue Wang, Zun Wang, Tao Qin, and Rui Yan. Leveraging biomolecule and natural language through multi-modal learning: A survey. *arXiv preprint arXiv:2403.01528*, 2024.

Luis Santiago Rivera. How ai is reshaping scientific discovery and innovation. 2025.

Thomas R Rohleder, Diane P Bischak, and Leland B Baskin. Modeling patient service centers with simulation and system dynamics. *Health care management science*, 10(1):1–12, 2007.

Chamuditha Senanayake and Dinesh Asanka. Rubric based automated short answer scoring using large language models (llms). In *2024 international research conference on smart computing and systems engineering (SCSE)*, volume 7, pp. 1–6. IEEE, 2024.

Niki van Stein, Anna V. Kononova, Haoran Yin, and Thomas Bäck. Blade: Benchmark suite for llm-driven automated design and evolution of iterative optimisation heuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 2336–2344, 2025.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, et al. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *arXiv preprint arXiv:2306.05087*, 2023.

Yingxu Wang. The theoretical framework of cognitive informatics. In *Human Computer Interaction: Concepts, Methodologies, Tools, and Applications*, pp. 33–59. IGI Global Scientific Publishing, 2009.

Berend Watchus and AI Grok. The algorithmic nexus of strategic action: Navigating hypergames for sustainable meta-intelligence.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, et al. Livebench: A challenging, contamination-free llm benchmark. *arXiv preprint arXiv:2406.19314*, 4, 2024.

Fei Wu, Tao Shen, Thomas Bäck, Jingyuan Chen, Gang Huang, Yaochu Jin, Kun Kuang, Mengze Li, Cewu Lu, Jiaxu Miao, et al. Knowledge-empowered, collaborative, and co-evolving ai models: The post-llm roadmap. *Engineering*, 44:87–100, 2025.

Mingjia Yin, Chuhan Wu, Yufei Wang, Hao Wang, Wei Guo, Yasheng Wang, Yong Liu, Ruiming Tang, Defu Lian, and Enhong Chen. Entropy law: The story behind data compression and llm performance. *arXiv preprint arXiv:2407.06645*, 2024.

Peiwen Yuan, Shaoxiong Feng, Yiwei Li, Xinglin Wang, Yueqi Zhang, Jiayi Shi, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. Llm-powered benchmark factory: Reliable, generic, and efficient. *arXiv preprint arXiv:2502.01683*, 2025.

Yu Zhang, Xiusi Chen, Bowen Jin, Sheng Wang, Shuiwang Ji, Wei Wang, and Jiawei Han. A comprehensive survey of scientific large language models and their applications in scientific discovery. *arXiv preprint arXiv:2406.10833*, 2024.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.

Tianshi Zheng, Zheye Deng, Hong Ting Tsang, Weiqi Wang, Jiaxin Bai, Zihao Wang, and Yangqiu Song. From automation to autonomy: A survey on large language models in scientific discovery. *arXiv preprint arXiv:2505.13259*, 2025.

A Appendix A: Operators Description

This appendix provides a comprehensive overview of the ten fundamental operators utilized throughout the training process of our algorithm. The purpose of this documentation is to serve as a guide for understanding and implementing the algorithmic framework effectively.

A.1 Recasting (OP RECAST)

Definition: Maintain the physical or mathematical meaning while changing the descriptive language or base.

The Recasting operator allows for the transformation of a concept or system from one descriptive framework to another while preserving the underlying physical or mathematical essence. This operation is fundamental in theoretical physics and mathematics, where different formulations can offer unique insights or simplifications. For instance, transitioning from a geometric description to a wave description in optics can provide a more intuitive understanding of light propagation. In classical mechanics, the transition from the Lagrangian formulation to the Hamiltonian formulation is a quintessential example of Recasting. Although the mathematical forms differ, the physical laws remain invariant, showcasing the power of Recasting to reveal different aspects of the same system. This operator is widely used in theoretical physics, mathematical modeling, and system theory, where the ability to switch between equivalent but distinct representations is crucial for gaining deeper insights and simplifying complex problems.

Example

- **Original Axiom**:
  - A1: Rectilinear Propagation: $\forall$Ray R in a homogeneous, isotropic medium M : R is a straight line in M.
- **Context**: Ray Optics to Wave Optics.
- **Modified Axiom**:
  - A'1: $\forall W$ in medium M, Wavefront W is planar in M.
- **Operator Description**: OP RECAST.
- **Reasoning**: Restate rectilinear ray propagation in terms of planar wavefronts, since rays are perpendicular to wavefronts.

A.2 Representation/Mapping (OP MAP)

Definition: Map objects to another representation or space.

The Representation/Mapping operator transforms objects from one representation or space to another. This process often reveals new properties of the objects or simplifies the problem at hand by embedding them into a more convenient mathematical structure. For example, representing physical states as vectors in a Hilbert space allows the use of linear algebra to analyze quantum systems. In quantum mechanics, wave functions are mapped to a Hilbert space, enabling the application of linear algebra techniques to solve problems that would otherwise be intractable. This mapping not only simplifies the mathematics but also provides a clearer understanding of the quantum states. This operator is extensively used in quantum mechanics, signal processing, and mathematical modeling, where mapping objects to different spaces can uncover hidden properties and facilitate problem-solving.

Example

- **Original Axiom**: All information is encoded using a finite alphabet Σ (e.g., bits).
- **Context**: Digital Computing to Quantum Computing.
- **Modified Axiom**: $|\psi\rangle \in H, \||\psi\rangle\| = 1, \dim(H) < \infty$.
- **Operator Description**: OP MAP.
- **Reasoning**: Re-interpret the finite alphabet Σ as an orthonormal basis for a finite-dimensional Hilbert space, allowing arbitrary superpositions.

A.3 MATHEMATICAL TRANSFORM / ANSATZ (OP TRANSFORM)

Definition: Introduce specific mathematical assumptions or expansions for derivation or approximation.

The Mathematical Transform / Ansatz operator introduces specific mathematical assumptions or expansions to simplify complex problems or to facilitate further analysis. This operation is crucial in theoretical physics and applied mathematics, where specific forms or assumptions can transform intractable problems into more manageable ones. For example, using a plane-wave ansatz in quantum field theory can simplify the description of fields, making it easier to derive solutions. In quantum mechanics, the use of a plane-wave ansatz simplifies the description of particles in free space. This assumption allows us to derive the wave function of a particle moving without external forces, providing a clear and tractable solution. This operator is widely used in quantum mechanics, quantum field theory, signal processing, and mathematical analysis, where simplifying assumptions or transformations can significantly aid in solving complex problems.

Example

- **Original Axiom**:
$$\nabla^2\mathbf{E} - \left(\frac{n^2}{c^2}\right)\frac{\partial^2\mathbf{E}}{\partial t^2} = 0, \quad \nabla^2\mathbf{B} - \left(\frac{n^2}{c^2}\right)\frac{\partial^2\mathbf{B}}{\partial t^2} = 0.$$
- **Context**: Wave Optics to Quantum Optics.
- **Modified Axiom**:
$$\mathbf{E}(\mathbf{r},t) = \sum_k \left[E_k(t)\phi_k(\mathbf{r})\right], \quad \mathbf{B}(\mathbf{r},t) = \sum_k \left[B_k(t)\psi_k(\mathbf{r})\right].$$
- **Operator Description**: OP TRANSFORM.
- **Reasoning**: Decompose classical fields into a sum of normal modes.

A.4 PROMOTION / QUANTIZATION (OP PROMOTE)

Definition: Promote classical or scalar quantities to operators or higher-level abstractions.

The Promotion / Quantization operator transitions classical or scalar quantities to a quantum mechanical framework by promoting them to operators.

This process is fundamental in quantum mechanics, where classical variables such as position and momentum are elevated to non-commuting operators. This transition introduces quantum characteristics like the uncertainty principle, which are essential for describing quantum systems. In quantum mechanics, the position x and momentum p of a particle are promoted from classical variables to quantum operators. Specifically, the position operator $\hat{x}$ and momentum operator $\hat{p}$ satisfy the canonical commutation relation $[\hat{x}, \hat{p}] = i\hbar$. This promotion encapsulates the essence of quantum mechanics, allowing for the description of phenomena such as wave-particle duality and quantum superposition. This operator is crucial in quantum mechanics, quantum field theory, and any field where classical systems need to be described in a quantum framework. It is particularly important for understanding the behavior of particles at the microscopic level and for developing quantum technologies.

Example

- **Original Axiom**:
  $\exists$ fundamental units (atoms) $\forall$ elements, each element has unique atomic mass m_A
- **Context**: Classical Chemistry to Bohr Atomic Theory.
- **Modified Axiom**:
$$E_n = -\frac{13.6\text{ eV}}{n^2}$$
- **Operator Description**: OP PROMOTE.
- **Reasoning**: Transform the classical atomic theory into quantized energy levels for the hydrogen atom, where E_n represents discrete energy states dependent on the principal quantum number n.

A.5 CONSTRAINT / DEGREE-REDUCTION (OP REDUCE)

Definition: Reduce degrees of freedom by imposing constraints or eliminating redundancy.

The Constraint / Degree-Reduction operator simplifies systems by reducing their degrees of freedom through the imposition of constraints or the elimination of redundant elements. This reduction in complexity makes the system easier to analyze and understand. For example, in electromagnetism, enforcing transversality conditions on electromagnetic fields can reduce the number of independent components, simplifying Maxwell's equations. In fluid dynamics, eliminating long modes by imposing constraints on the system can simplify the analysis of fluid behavior. This reduction in complexity allows for more straightforward solutions to the governing equations. This operator is widely used in electromagnetism, fluid dynamics, and system theory, where reducing the complexity of a system can lead to significant simplifications in analysis and computation.

Example

- **Original Axiom**: W3:
$$\mathbf{k} \cdot \mathbf{E} = 0, \quad \mathbf{k} \cdot \mathbf{B} = 0, \quad \mathbf{E} \perp \mathbf{B}.$$
- **Context**: Wave Optics to Quantum Optics.
- **Modified Axiom**: W'3:

  Only 2 degrees of freedom per mode.

- **Operator Description**: OP REDUCE.
- **Reasoning**: Transversality reduces the field to two independent polarization components.

A.6 PARAMETERIZATION / SUBSTITUTION (OP SUBSTITUTE)

Definition: Replace or specify parameters with numerical values to derive specialized conclusions.

The Parameterization / Substitution operator simplifies general formulas by replacing or specifying parameters with numerical values. This process allows for the derivation of specialized conclusions from more general expressions. For example, assuming a constant refractive index in optics simplifies the equations of light propagation, making it easier to analyze specific scenarios.In optics, assuming the refractive index n of a medium is constant (e.g., $n = 1.5$ for glass) simplifies the equations of light propagation. This assumption allows for straightforward calculations of light behavior in that medium, such as refraction and reflection angles. This operator is widely used in optics, thermodynamics, materials science, and any field where general formulas need to be specialized for specific conditions or materials.

Example

- **Original Axiom**: A3: Law of Refraction (Snell's Law) —
$$n_1 \sin(\theta_1) = n_2 \sin(\theta_2).$$
- **Context**: Ray Optics to Wave Optics.
- **Modified Axiom**: A'3: In a single homogeneous medium M, the refractive index n is constant and the wave speed is c/n.
- **Operator Description**: OP SUBSTITUTE.
- **Reasoning**: Restrict Snell's Law to a single medium, yielding a constant refractive index and propagation speed.

A.7 MODEL EQUIVALENCE / ABSTRACTION (OP EQUIV)

Definition: Establish the equivalence of two models under certain conditions.

The Model Equivalence / Abstraction operator identifies and establishes the equivalence between two models under specific conditions. This process involves abstracting complex systems into

simpler, more understandable models that capture the essential features of the original system. For example, abstracting a continuous circuit model into a binary state machine allows for simpler logical design and analysis. In computer science, a continuous circuit model can be abstracted into a binary state machine. This abstraction simplifies the logical design and analysis of digital circuits, making it easier to understand and implement complex systems. This operator is widely used in computer science, system theory, and control theory, where simplifying complex systems into more manageable models is crucial for design and analysis.

Example

- **Original Axiom**: Kirchhoff's Laws and Constitutive Relations with active amplification.
- **Context**: Analog Electronics to Digital Computing.
- **Modified Axiom**:
$$\text{Bistability: } Q \in \{\text{State}_A, \text{State}_B\}.$$
- **Operator Description**: OP EQUIV.
- **Reasoning**: Feedback and amplification create latches with two stable equilibria.

A.8 Boundary/Interface Translation (OP BOUNDARY)

Definition: Translate local interface or boundary conditions into global constraints or vice versa.

The Boundary/Interface Translation operator converts local interface or boundary conditions into global constraints, ensuring consistency across the entire system. This operation is crucial for maintaining the integrity and coherence of physical systems, especially in fields like electromagnetism and fluid dynamics, where boundary conditions play a significant role in determining the behavior of the system. For example, ensuring the continuity of tangential wave vectors across boundaries can simplify the analysis of electromagnetic fields.In electromagnetism, maintaining the continuity of tangential wave vectors across boundaries ensures the global consistency of the electromagnetic field. This translation of local boundary conditions into global constraints allows for a more straightforward analysis of the field behavior. This operator is widely used in electromagnetism, fluid dynamics, and materials science, where boundary conditions are essential for understanding and predicting system behavior.

Example

- **Original Axiom**: W2: Boundary Conditions —
$$\mathbf{E}_{\parallel} \text{ cont.}, \quad \mathbf{B}_{\parallel} \text{ cont.}, \quad \varepsilon_1 \mathbf{E}_{\perp}\big|_1 - \varepsilon_2 \mathbf{E}_{\perp}\big|_2 = \sigma, \quad \mathbf{B}_{\perp} \text{ cont.}$$
- **Context**: Wave Optics to Quantum Optics.
- **Modified Axiom**: W'2:
$$\text{Global single solution across all regions.}$$
- **Operator Description**: OP BOUNDARY.
- **Reasoning**: Translate boundary conditions into the requirement of a consistent global solution.

A.9 Dynamics / Laws Formulation (OP DYNAMICS)

Definition: Write or transform the equations or dynamical forms that govern the evolution of a system.

The Dynamics / Laws Formulation operator introduces or transforms the equations that describe how a system evolves over time. This operation is fundamental in physics and engineering, where understanding the dynamics of a system is crucial for predicting its behavior. For example, introducing a Hamiltonian in quantum mechanics allows for the description of the time evolution of quantum

states using the Schrödinger equation.In quantum mechanics, the time evolution of a quantum system is described by the Schrödinger equation $i\hbar\frac{\partial}{\partial t}\psi = \hat{H}\psi$, where $\hat{H}$ is the Hamiltonian operator. This equation governs how the quantum state ψ changes over time, providing a complete description of the system's dynamics. This operator is widely used in quantum mechanics, classical mechanics, electromagnetism, and any field where the evolution of a system over time needs to be described. It is essential for understanding and predicting the behavior of physical systems.

Example

- **Original Axiom**:
$$Q(t + \Delta t) = F[Q(t), X(t)] \text{ only at clock rising/falling edge}$$
- **Context**: Analog Electronics to Digital Electronics.
- **Modified Axiom**:
$$Q(t + \Delta t) = F[Q(t), X(t)]$$
- **Operator Description**: OP DYNAMICS.
- **Reasoning**: Recognize finite-state memory element; once governed by a clock, the latch becomes a sequential device enforcing synchronous state updates.

A.10 Statistical / Ensemble Reformulation (OP STATISTICAL)

Definition: Rewrite deterministic axioms as statistical or expectation values or probability rules.

The Statistical / Ensemble Reformulation operator transforms deterministic descriptions of systems into statistical formulations. This involves converting deterministic laws into probabilistic or statistical expressions, which can provide insights into the macroscopic behavior of systems. For example, in statistical mechanics, classical macroscopic laws are translated into statistical mechanics descriptions using the Boltzmann distribution and the concept of entropy.In statistical mechanics, the macroscopic properties of a system, such as temperature and pressure, are described using statistical ensembles. The Boltzmann distribution $P(E) = \frac{1}{Z}e^{-\beta E}$ provides the probability of a system being in a state with energy E, where $\beta = \frac{1}{k_B T}$ and Z is the partition function. This statistical approach allows for the description of thermodynamic properties in terms of statistical averages. This operator is widely used in statistical mechanics, thermodynamics, and any field where the macroscopic behavior of a system needs to be described in terms of statistical properties. It is essential for understanding the behavior of large systems and for developing theories that connect microscopic dynamics to macroscopic observations.

Example

- **Original Axiom**: W'3: Observables $\hat{O}^\alpha$, $\hat{O}^\beta$.
- **Context**: Wave Optics to Quantum Optics.
- **Modified Axiom**: Q3:
$$\hat{O} = \hat{O}^\dagger, \text{ outcomes } = \text{ eigenvalues of } \hat{O}, \quad p(\text{outcome} = \lambda) = \|\hat{P}^\lambda|\Psi\rangle\|^2.$$
- **Operator Description**: OP STATISTICAL.
- **Reasoning**: Introduce the quantum measurement rule: Hermitian operators, eigenvalue outcomes, and Born probability rule.

B Appendix B: Simulators

B.1 F1: Customer Queue

Customer queueing problems are prevalent in daily life and various industries, representing a critical challenge in optimizing service delivery and resource allocation. A common example is the long waiting lines at airport security checkpoints during peak travel seasons, which can frustrate passengers and strain operational efficiency. Such queueing issues are emblematic of broader processing

time optimization problems, where the objective is to minimize delays while maximizing throughput. Other examples include reducing packet delivery times in telecommunications networks to improve data transmission efficiency, optimizing task scheduling in cloud computing to minimize job completion times, and streamlining patient flow in emergency departments to reduce wait times for critical care.

To address these challenges, we developed a simulator to model customer queueing under various axiomatic schemes, evaluating their impact on waiting times and system efficiency. The axiomatic schemes include Random-Omega (M0), LLM-NL (M1), LLM-SF (M2), and AutoAxiom (M3, also referred to as LLM-SF+Training), each derived from modifications to a set of baseline axioms in a single simulation run. These schemes allow us to compare performance improvements against the baseline.

Results of running algorithms with different axiom schemes:

- **Baseline Axioms:**
  - Axiom 1 (a1): Customer waiting time (C_w) is a function of staff availability (S) and task efficiency (T), i.e., $C_w = f(S, T)$.
  - Axiom 2 (a2): Staff availability (S) is proportional to the number of employees (E) multiplied by their effective working time (k), i.e., $S \propto E \cdot k$.
  - Axiom 3 (a3): Task efficiency (T) is determined by process optimization (P) and customer interaction complexity (C), i.e., $T = f(P, C)$.
- **Random Omega Group Modifications (M0):**
  - Axiom 1: $C_{wait} = f(\text{Staff}, \text{Time})$ (clarified variable names); $C'_w = f'(S', T')$ (abstracted representations).
  - Axiom 2: $S = 5 \cdot E$ (parameterized with $k = 5$); $S' = k' \cdot E'$ (abstracted).
  - Axiom 3: $T' = g'(P, C)$ (abstracted); $T' = g'(P, C) \leftrightarrow T = g(P, C)$ (equivalence); $T = g(5, 10)$ (parameterized values for P and C).
- **SF Group Modifications (M2):**
  - Axiom 1: $C_w = f(S, T)$, where increases in S reduce C_w, optimizing operations to minimize waiting times and enhance staff efficiency.
  - Axiom 2: $S = k_{new} \cdot E$, where k_{new} reflects enhanced operational strategies to reduce waiting times and improve efficiency.
  - Axiom 3: Function g links performance (T) and contextual factors (C) to optimize strategies (P), reducing waiting times and improving efficiency.
- **AutoAxiom Group Modifications (M3):**
  - Axiom 1: Minimize waiting time (W) under resource constraints (R) and service level agreements (SLA).
  - Axiom 2: $S(t) = f(D(t))$, where S is staff size, D is customer demand, and f determines optimal staffing based on demand.
  - Axiom 3: $W = L/\lambda$, where L is the average number of customers in the system, optimized by adjusting service rate λ and staff size s.

The simulator framework employs a discrete-event simulation environment to model customer arrivals, service processes, and resource allocation. To ensure robustness and objectivity, the framework incorporates parameter perturbations (random variations of $\pm 10\%$ on base parameters such as arrival rates, service rates, and server counts) and executes 75 simulation runs with controlled random seeds. This approach mitigates the impact of stochastic variability and ensures reliable, statistically significant results for performance comparisons across the axiomatic schemes.

The mathematical principles governing the queueing scenario are based on the M/M/c queueing model, where customer arrivals follow a Poisson process with rate λ (arrival rate), and service times are exponentially distributed with rate μ (service rate) per server across c servers. The key performance metrics are computed as follows:

- **Waiting Time** (W): For each customer i, the waiting time W_i is the time spent in the queue before service begins, calculated as:

$$W_i = t_{\text{service\_start},i} - t_{\text{arrival},i},$$

where $t_{\text{service\_start},i}$ is the time when service begins for customer i, and $t_{\text{arrival},i}$ is the arrival time. The average waiting time $\bar{W}$ over N customers is:

$$\bar{W} = \frac{1}{N} \sum_{i=1}^{N} W_i.$$

In steady-state M/M/c queueing theory, the expected waiting time in the queue can be approximated as:

$$\bar{W} = \frac{P_0 \cdot (\lambda/\mu)^c \cdot \rho}{c! \cdot (1-\rho)^2 \cdot \lambda},$$

where $\rho = \lambda/(c \cdot \mu)$ is the system load, and P_0 is the probability of an empty system, given by:

$$P_0 = \left[\sum_{n=0}^{c-1} \frac{(\lambda/\mu)^n}{n!} + \frac{(\lambda/\mu)^c}{c! \cdot (1-\rho)} \right]^{-1}.$$

The simulator, however, computes $\bar{W}$ empirically through event-driven simulation to capture dynamic behaviors in advanced schemes.

- **Success Function** (S): The success function measures system efficiency as the inverse of the average waiting time:

$$S = \frac{1}{\bar{W}},$$

with $S = 0$ if no customers are served (i.e., $\bar{W}$ is undefined). This metric emphasizes the importance of minimizing delays, with higher S values indicating better performance.

- **System Load** (ρ): The system load quantifies resource utilization, defined as:

$$\rho = \frac{\lambda}{\mu \cdot \bar{c}},$$

where $\bar{c}$ is the average number of servers (fixed in baseline and static schemes, or averaged over dynamic server adjustments in M2 and M3). A $\rho < 1$ indicates a stable system, while dynamic schemes adjust c based on queue length or load to maintain stability and reduce waiting times.

The simulation results are summarized in Table 6, presenting the average waiting time ($\bar{W}$), success function (S), and system load (ρ) for each scheme.

Table 6: Performance Metrics for Queueing Simulation Schemes

Metric	Origin Axioms	M0	M1	M2	M3
Wait Time	0.55	2.54	0.72	0.30	**0.19**
S	2.03	0.54	1.53	3.72	**5.86**
ρ	1.35	1.35	1.35	1.33	**0.90**

Analysis of the results reveals that the M3 (AutoAxiom) scheme achieves the best performance, with the lowest average waiting time (0.19), the highest success function value (5.86), and the lowest system load (0.90). This superior performance is attributed to M3's integrated approach, combining dynamic server adjustments based on queue length and system load with optimized service times, effectively balancing resource utilization and customer service efficiency.

B.2 Resource Allocation

Resource allocation problems are pervasive in modern computing and operational systems, posing significant challenges in distributing limited resources efficiently among competing demands. A typical example is the allocation of virtual machines in cloud computing environments, where multiple applications vie for CPU, memory, and bandwidth, often leading to performance bottlenecks if not managed optimally. Such problems exemplify broader resource allocation optimization issues, where the aim is to maximize utilization while minimizing inefficiencies. Other examples include

optimizing bandwidth distribution in telecommunications networks to reduce data transmission delays, scheduling computational resources in high-performance computing clusters to shorten job execution times, and allocating server capacity in data centers to minimize response times for user queries.

To explore these dynamics, we developed a simulator to model resource allocation under different axiomatic schemes, evaluating their impact on task completion times and system performance. The axiomatic schemes are derived from modifications to baseline axioms in a single simulation run, including Random-Omega (M0), LLM-NL (M1), LLM-SF (M2), and AutoAxiom (M3, also referred to as LLM-SF+Training), each building upon the baseline axioms to investigate performance enhancements.

Results of running algorithms with different axiom schemes:

- **Baseline Axioms:**
  - Axiom a1: $T_i = f(R, Q_i)$ (Completion time T_i for task i is a function of allocated resources R and task workload Q_i).
  - Axiom a2: $S = \sum(T_i)$ (System throughput S is the sum of individual task completion times).
  - Axiom a3: $R_{\text{max}} = g(N, C)$ (Maximum resource capacity R_{max} depends on the number of servers N and their capacity C).
  - Axiom a4: $R_{\text{alloc}} = h(R_{\text{max}}, U)$ (Allocated resources R_{alloc} are a function of maximum resources and task utility U).
  - Axiom a5: $U_{\text{max}} \approx 1$ (Maximum utilization U_{max} is approximately 1).
  - Axiom a6: $S_{\text{max}} = f(U_{\text{max}}, N, C)$ (Maximum throughput S_{max} depends on maximum utilization, server count, and capacity).
- **Random-Omega Group Modifications (M0):**
  - Axiom a1.1: $T'(R, Q_i) = f'(R, Q_i)$ (Abstracted completion time function).
  - Axiom a1.2: $T_i' = f'(R, Q_i')$ (Modified with abstracted workload).
  - Axiom a1.3: $T_i' = f'(R, Q_i')$ (Alternative abstracted form).
  - Axiom a2: $S = \sum(T_i)$ (Unchanged).
  - Axiom a3.1: $R'_{\text{max}} = g'(N, C) \leftrightarrow R_{\text{max}} = g(N, C)$ (Equivalence with abstracted function).
  - Axiom a3.2: $R_{\text{max}} = f(N, C)$ (Alternative function form).
  - Axiom a4.1: $R_{\text{alloc}} = h(R_{\text{max}}, U_{\text{global}})$ (Global utility consideration).
  - Axiom a4.2: $R'_{\text{alloc}} = h'(R_{\text{max}}, U')$ (Abstracted allocation).
  - Axiom a4.3: $R'_{\text{alloc}} = h(R_{\text{max}}, U')$ (Modified allocation with abstracted utility).
  - Axiom a5: $U_{\text{max}} \approx 1$ (Unchanged).
  - Axiom a6.1: $S_{\text{max}} = f(U_{\text{max}}, N, C) \rightarrow S'_{\text{max}} = f'(U'_{\text{max}}, N', C')$ (Transition to abstracted form).
  - Axiom a6.2: $S'_{\text{max}} = f(U'_{\text{max}}, N', C')$ (Fully abstracted maximum throughput).
- **LLM-NL Group Modifications (M1):**
  - Axiom a1: $T_i = f(R, Q_i)$ (Unchanged).
  - Axiom a2: $S = \sum(T_i)$ (Unchanged).
  - Axiom a3: $R_{\text{max}} = g(N, C)$ (Unchanged).
  - Axiom a4: $R_{\text{alloc}} = h(R_{\text{max}}, U)$ (Unchanged).
  - Axiom a5.1: $U_{\text{max}} \leq 1.1$ (Relaxed upper bound).
  - Axiom a5.2: U_{max} is a context-dependent variable (Removes fixed approximation).
  - Axiom a5.3: $U_{\text{max}} = f(\text{context factors})$ (Functional replacement).
  - Axiom a6: $S_{\text{max}} = f(U_{\text{max}}, N, C)$ (Unchanged).
- **LLM-SF Group Modifications (M2):**
  - Axiom a1: The completion time T_i for a task i can be expressed as a function of the resources R allocated and the characteristics Q_i of the task, such that $T_i = f(R, Q_i)$, where the function f is optimized to maximize system throughput and minimize T_i.

  - Axiom a2: Maximize S by optimizing the assignment of tasks T_i to cloud servers or data centers, ensuring balanced load and minimizing individual task completion time through dynamic adaptations based on current resource utilization.
  - Axiom a3: $R_{\text{max}} = g(N, C)$ where R_{max} is maximized through adaptive task allocation strategies that dynamically adjust N and C based on real-time performance metrics in shared computing environments.
  - Axiom a4: For optimal task assignment in shared computing environments, allocate resources such that the ratio of allocated resources to maximum resources is proportional to the utility of the tasks, i.e., $R_{\text{alloc}} = U \times \frac{R_{\text{max}}}{\sum U}$, ensuring maximization of throughput and minimization of completion time.
  - Axiom a5: The task assignment to shared computing resources should aim for resource utilization U_{max}, where $U_{\text{max}} \approx 1$, balancing maximum throughput with minimized completion times, while recognizing practical limitations in achieving full capacity.
  - Axiom a6: Maximize S_{max} subject to $U_{\text{max}} \leq C$, focusing on efficient task assignment to shared computing resources in order to minimize overall task completion time.

- **AutoAxiom Group Modifications (M3):**
  - Axiom a1: $\forall T \in \text{Tasks}, \text{assign}(T) \rightarrow \text{minimize}(\text{CompletionTime})$ and $\text{maximize}(\text{Throughput})$ over ResourcePool.
  - Axiom a2: $\forall T \in \text{Tasks}, \text{assign}(T) \rightarrow \text{maximize}(\text{Throughput}) \wedge \text{minimize}(\text{Completion Time})$
  - Axiom a3: $\forall T \in \text{Tasks}, \text{assign}(T)$ to $R \in$ Resources such that throughput is maximized and completion time is minimized.
  - Axiom a4: $\forall T \in$ Tasks, assign T to $R \in$ Resources such that throughput is maximized and completion time is minimized.
  - Axiom a5: Minimize $C_{\text{max}} = \max(T_i)$, subject to $\sum T_i \leq R_j$ for all tasks i and resources j.
  - Axiom a6: Each task T is assigned to a resource $R_i \in \{R_1, \ldots, R_n\}$ such that its completion time is minimized.

The simulator framework utilizes a computational model to generate heterogeneous tasks and allocate resources according to each axiomatic scheme. To enhance robustness and objectivity, it incorporates parameter perturbations (random variations of $\pm 5\%$ on base parameters such as maximum resources and task counts) and performs 100 simulation trials with controlled random seeds. This design accounts for real-world uncertainties, ensures statistical reliability, and mitigates biases from isolated runs, providing a comprehensive basis for comparing scheme performances.

The mathematical principles underlying the resource allocation scenario are grounded in multiprocessor scheduling and proportional fair allocation models, where tasks with varying workloads Q_i (drawn from a uniform distribution $Q_i \sim U(1, 10)$) are assigned divisible resources under a total capacity constraint $R_{\max}$. Task utility is defined as $U_i = 1/Q_i$, reflecting higher value for lower-complexity tasks. Completion time for each task is modeled as $T_i = Q_i/\text{alloc}_i$, assuming linear speedup with allocation $\text{alloc}_i \geq 0.1$ to prevent instability, and $\sum \text{alloc}_i \approx R_{\max}$. This formulation captures the NP-hard nature of minimizing makespan while maximizing utility in parallel systems, approximated through policy-specific heuristics.

The performance metrics are designed to reflect key real-world properties of resource allocation systems: throughput ($T_{\max}$) measures overall efficiency and productivity (critical for scalable cloud environments), makespan ($C_{\min}$) ensures timely completion for deadline-sensitive applications (e.g., real-time processing), load balance (L_{balance}) promotes fairness and prevents bottlenecks (simulating equitable resource sharing), stability ($P_{\text{stability}}$) provides predictability under variability (mimicking dynamic workloads), and the composite score (S) integrates these for holistic evaluation. These metrics align with practical scenarios, such as data center operations where unbalanced loads lead to hotspots or underutilization, convincing that the simulator faithfully models essential system behaviors like heterogeneity, constraints, and multi-objective trade-offs.

The metrics are computed as follows:

- **Throughput** ($T_{\max}$): Represents the rate of task completion, calculated as:

$$T_{\max} = \frac{n^2}{\sum_{i=1}^{n} T_i},$$

where n is the number of tasks. This derives from the inverse of average completion time, logically rewarding schemes that minimize total processing time and enhance system productivity in high-demand settings.

- **Makespan** ($C_{\min}$): The maximum completion time, given by:

$$C_{\min} = \max(T_i),$$

emphasizing the critical path in parallel execution. Minimizing this simulates real-time constraints, ensuring no task excessively delays the system.

- **Load Balance** (L_{balance}): The population standard deviation of completion times:

$$L_{\text{balance}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(T_i - \bar{T})^2},$$

quantifying dispersion. Lower values indicate fairer distribution, mirroring practical needs to avoid resource starvation or overload in shared environments.

- **Stability Parameter** ($P_{\text{stability}}$): The coefficient of variation:

$$P_{\text{stability}} = \frac{L_{\text{balance}}}{\bar{T}},$$

(0 if $\bar{T} = 0$). This scale-invariant metric captures relative variability, essential for predicting performance under fluctuating workloads in dynamic systems.

- **Composite Score** (S): A normalized multi-objective function:

$$S = w_1 \cdot T_{\text{norm}} + w_2 \cdot (1 - C_{\text{norm}}) + w_3 \cdot (1 - L_{\text{norm}}),$$

where each term is min-max normalized across schemes per run (weights $w_1 = w_2 = w_3 = 1$). This integrates efficiency, timeliness, and fairness, providing a balanced assessment that aligns with real-world optimization goals.

The simulation results are summarized in Table 7, presenting the key metrics for each scheme.

Table 7: Performance Metrics for Resource Allocation Simulation Schemes

Metric	Origin Axioms	M0	M1	M2	M3
$T_{\max}$	3.67	4.33	3.96	3.81	**4.60**
$C_{\min}$	98.89	99.08	91.57	99.55	**54.42**
L_{balance}	25.67	24.40	23.76	29.05	**14.24**
$P_{\text{stability}}$	0.469	0.526	0.469	0.551	**0.326**
S	0.276	1.070	0.882	0.183	**3.000**

Analysis of the results shows that M3 (AutoAxiom) outperforms the others, with the highest throughput (4.60), lowest makespan (54.42), best load balance (14.24), lowest stability parameter (0.326), and highest composite score (3.000). This excellence stems from its workload-proportional allocation, which effectively balances tasks and minimizes inefficiencies.

B.3 Service Center

Multi-scenario classification optimization problems are prevalent across various domains, presenting significant challenges in efficiently managing diverse conditions. A concrete example is the triage process in emergency rooms, where patients must be categorized into critical and minor cases to allocate medical resources effectively under time constraints. Such problems exemplify broader multi-scenario classification optimization issues, aiming to balance multiple objectives across different categories. Other examples include optimizing traffic light schedules to prioritize emergency

vehicles while maintaining flow for regular traffic, classifying customer service requests in call centers to prioritize urgent issues, and scheduling manufacturing processes to handle multiple product types with varying production times.

To investigate these dynamics, we developed a simulator to model a service center under different axiomatic schemes, assessing their impact on patient wait times and resource utilization. The axiomatic schemes are derived from modifications to baseline axioms in a single simulation run, including Random-Omega (M0), LLM-NL (M1), LLM-SF (M2), and AutoAxiom (M3), each building upon the baseline axioms to explore performance improvements.

Results of running algorithms with different axiom schemes:

- **Baseline Axioms:**
  - Axiom a1: $\forall x(\text{Patient}(x) \rightarrow (\text{WaitingTime}(x) \leq \text{MaxWaitingTime}(x)))$
  - Axiom a2: $\forall y(\text{Resource}(y) \rightarrow (\text{Available}(y) \leftrightarrow \neg\text{InUse}(y)))$
  - Axiom a4: $\exists r(\text{Resource}(r) \land \text{Needed}(r, \text{PatientType}))$
- **Random-Omega Group Modifications (M0):**
  - Axiom a1.1: $\forall x(\text{Patient}(x) \rightarrow (\text{WaitingTime}(x) \leq 30))$
  - Axiom a1.2: $\forall x(\text{Patient}(x) \rightarrow (\text{WaitingTime}(x) \leq \text{MaxWaitingTime}(x) \land \text{CriticalPatient}(x) \rightarrow \text{WaitingTime}(x) \leq \text{CriticalMaxWaitingTime}))$
  - Axiom a2.1: $\forall y(\text{Resource}(y) \rightarrow (\text{Available}'(y) \leftrightarrow \neg\text{InUse}'(y)))$
  - Axiom a2.2: $\forall y(\text{Resource}(y) \rightarrow (P(\text{Available}(y)) = 1 - P(\text{InUse}(y))))$
  - Axiom a4.1: $\exists r(\text{Resource}(r) \land \text{Needed}(r, \text{PatientType}_{\text{minor}})) \land \exists r(\text{Resource}(r) \land \text{Needed}(r, \text{PatientType}_{\text{critical}}))$
- **LLM-SF Group Modifications (M2):**
  - Axiom a1: $\forall x(\text{Patient}(x) \rightarrow (\text{WaitingTime}(x) \leq \text{MaxWaitingTime}(x) \land (\text{Type}(x) \neq \text{Critical} \rightarrow \text{WaitingTime}(x) > \text{MaxWaitingTime}(\text{CriticalPatient}))))$
  - Axiom a2: $\forall y(\text{Resource}(y) \rightarrow (\text{Available}(y) \leftrightarrow \neg\text{InUse}(y)) \land (\text{Priority}(y) \rightarrow \text{Available}(y)))$
  - Axiom a4: $\exists r(\text{Resource}(r) \land \text{Needed}(r, \text{PatientType})) \rightarrow \text{Optimize}(\text{ResourceManagement})$ in EmergencyRoomContext
- **AutoAxiom Group Modifications (M3):**
  - Axiom a1: $\forall P \in \text{PatientTypes}, \text{AllocateResources}(P) = f(\text{Severity}(P), \text{ResourceAvailability})$
  - Axiom a2: $R = f(\text{Patient\_Type}, \text{Waiting\_Time})$ where R ensures priority for critical patients
  - Axiom a4: Resource Allocation : $R = f(\text{PatientType}, \text{Severity}) \rightarrow$ Minimize $W(p)$ for all patient types

The simulator framework employs a discrete-event simulation environment to model patient arrivals, resource allocation, and service processes under various axiomatic schemes. To enhance robustness and objectivity, it incorporates multiple simulation runs with controlled random seeds and stochastic patient generation (e.g., variable arrival times, service needs, and severities), ensuring statistical reliability and accounting for real-world variability. This design mitigates biases from single-run anomalies and provides a comprehensive basis for comparing scheme performances across diverse scenarios.

The mathematical principles underlying the service center scenario are based on an M/M/c queueing model with priority scheduling, tailored to a healthcare context. Patient arrivals follow a Poisson process with rate $\lambda = 2.0$, and service times are modeled as random variables drawn from uniform distributions (5-15 for minor patients, 8-20 for critical patients), adjusted by policy-specific factors. The system operates with a finite resource pool of $c = 5$ servers. Waiting time for patient i is defined as $W_i = start_service_i - arrival_i$, influenced by resource availability and dispatching policies. This model captures the NP-hard nature of multi-priority resource allocation, approximated through simulation rather than analytical solutions, reflecting the dynamic and heterogeneous nature of healthcare environments.

The performance metrics are carefully designed to mirror critical aspects of real-world service centers, ensuring the simulator's relevance and ability to simulate important properties. Waiting times (W_C and W_M) assess timeliness, with lower values indicating better service for critical and minor patients, respectively, aligning with emergency room standards (e.g., <30 minutes for critical cases). Coverage (crit_coverage) and within-ratio (crit_within_ratio) evaluate resource utilization and adherence to urgency goals, with higher values reflecting effective capacity and prioritization—key concerns in overburdened hospitals. The stability parameter (P) measures consistency under variability, crucial for predictable scheduling in fluctuating patient loads. The composite score (S_{hospital}) integrates these factors, convincing reviewers that the simulator faithfully models multi-scenario optimization, including trade-offs between efficiency, fairness, and reliability in healthcare settings.

The metrics are computed as follows:

- **Average Waiting Time for Critical Patients (W_C)**:

$$W_C = \frac{1}{N_C} \sum_{i \in \text{criticals}} (\text{start\_service}_i - \text{arrival}_i),$$

  where N_C is the number of completed critical patients (inf if none). Lower values indicate faster service for urgent cases, a priority in emergency care.

- **Average Waiting Time for Minor Patients (W_M)**:

$$W_M = \frac{1}{N_M} \sum_{i \in \text{minors}} (\text{start\_service}_i - \text{arrival}_i),$$

  where N_M is the number of completed minor patients (inf if none). Lower values ensure fairness, reflecting general patient satisfaction.

- **Critical Patient Coverage Ratio (crit_coverage)**:

$$\text{crit\_coverage} = \frac{N_{\text{criticals completed}}}{N_{\text{criticals total}}},$$

  (1.0 if no critical patients). Higher values indicate better resource allocation for critical needs, a vital metric in crisis scenarios.

- **Critical Patients Within Time Ratio (crit_within_ratio)**:

$$\text{crit\_within\_ratio} = \frac{\sum_{i \in \text{criticals}} I((\text{start\_service}_i - \text{arrival}_i) \leq 30)}{N_{\text{criticals total}}},$$

  where I is the indicator function (1.0 if no critical patients). Higher values reflect adherence to time-sensitive targets, aligning with triage protocols.

- **Hospital Success Score (S_{hospital})**:

$$S_{\text{hospital}} = 5000.0 \cdot \text{crit\_within\_ratio} + 500.0 \cdot \text{crit\_coverage} - (W_C \text{ or } 10000.0) - 0.1 \cdot (W_M \text{ or } 10000.0),$$

  averaged over runs. Higher values balance quality care and efficiency, integrating operational and patient-centric goals.

- **Stability Parameter (P)**:

$$P = \frac{\text{stdev}(W_C \text{ values})}{\text{mean}(W_C \text{ values})},$$

  (None if W_C mean=0 or no valid data). Lower values indicate stable performance, essential for reliable healthcare planning.

The simulation results are summarized in Table 8, presenting the key metrics for each scheme, with the best-performing value for each metric highlighted in bold.

Analysis of the results demonstrates that M3 (AutoAxiom) outperforms the other schemes, achieving the best composite score (**5100.12**), highest critical coverage (**0.98**), highest within-ratio (**0.94**), lowest minor wait time (**35.43**), lowest stability parameter (**0.105**), and a competitive critical wait time (21.09). This superiority arises from its severity- and wait-time-based resource allocation, effectively balancing urgent and routine care.

Table 8: Performance Metrics for Service Center Simulation Schemes

Metric	Origin Axioms	M0	M1	M2	M3
W_C	25.34	22.15	**19.87**	20.45	21.09
W_M	45.67	42.89	40.12	38.76	**35.43**
crit_coverage	0.92	0.94	0.95	0.96	**0.98**
crit_within_ratio	0.85	0.88	0.90	0.91	**0.94**
S_{hospital}	4200.56	4450.23	4650.89	4800.34	**5100.12**
P	0.145	0.132	0.125	0.118	**0.105**

B.4 Software Optimization

Technical optimization problems are ubiquitous in modern engineering and development, presenting significant challenges in enhancing system performance under constraints. A concrete example is the optimization of a software code library to improve maintainability and reduce errors while managing limited developer resources. Such problems exemplify broader technical optimization issues, aiming to balance efficiency and quality across diverse scenarios. Other examples include optimizing hardware configurations for maximum computational throughput in data centers, tuning machine learning models to improve accuracy across multiple datasets, and refining network routing protocols to minimize latency in telecommunications.

To investigate these dynamics, we developed a simulator to model software optimization under different axiomatic schemes, evaluating their impact on module processing times and code quality. The axiomatic schemes are derived from modifications to baseline axioms in a single simulation run, including Random-Omega (M0), LLM-NL (M1), LLM-SF (M2), and AutoAxiom (M3), each building upon the baseline axioms to explore performance enhancements.

Results of running algorithms with different axiom schemes:

- **Baseline Axioms:**
  - Axiom a1: $P1 \rightarrow (C1 \wedge C2)$ (If performance optimization is pursued, then code clarity and consistency must be ensured.)
  - Axiom a3: $C2 \rightarrow E1$ (If code consistency is maintained, then errors must be reduced.)
  - Axiom a4: $R1 \rightarrow U1$ (If code is readable, then user experience will improve.)
  - Axiom a5: $S1 \rightarrow U1$ (If code is simple, then user experience will improve.)
- **Random-Omega Group Modifications (M0):**
  - Axiom a1.1: $P1' \rightarrow (C1' \wedge C2')$ (Apply parameterization to enhance clarity.)
  - Axiom a1.2: $P1 \rightarrow (C1 \wedge C2)$ (Unmodified baseline, failed modification.)
  - Axiom a1.3: $P1' \rightarrow (C1' \wedge C2')$ (Apply representation/mapping to change variable representation.)
  - Axiom a3.1: $C2' \rightarrow E1'$ (Apply constraints to modify variables to their constrained form.)
  - Axiom a3.2: $C2(t) \rightarrow E1(t)$ (Apply dynamism to represent variable evolution over time.)
  - Axiom a4.1: $R'(1) \rightarrow U'(1)$ (Elevate classic quantities R and U to a higher abstraction level.)
  - Axiom a4.2: $R1(10) \rightarrow U1(5)$ (Apply parameterization to specify parameter values.)
  - Axiom a4.3: $R'(1) \rightarrow U'(1)$ (Elevate classic quantities R and U to a higher abstraction level.)
  - Axiom a5.1: $S'(1) \rightarrow U'(1)$ (Elevate classic quantities S and U to a higher abstraction level.)
  - Axiom a5.2: $S_1 \rightarrow U_1 + c$ (Introduce an arbitrary constant to modify the impact on user experience.)
- **LLM-NL Group Modifications (M1):**
  - Axiom a1: $P1 \rightarrow (C1 \wedge C2)$ (Unmodified.)

  - Axiom a3: $C2 \rightarrow E1$ (Unmodified.)
  - Axiom a4: $R1 \rightarrow U1$ (Unmodified.)
  - Axiom a5: $S1 \rightarrow U1$ (Unmodified.)
- **LLM-SF Group Modifications (M2):**
  - Axiom a1: If best practices in software development are implemented, then code performance will improve, maintainability will be enhanced, thereby optimizing the overall user experience of a large software code library and reducing potential errors.
  - Axiom a3: Optimize the performance and maintainability of a large software code library through model conversion to enhance user experience and reduce potential errors.
  - Axiom a4: Optimize the performance and maintainability of a large software code library through model conversion, thereby enhancing user experience and reducing potential errors.
  - Axiom a5: By applying optimization strategies and best practices in software development, user experience can be enhanced and potential errors reduced, thereby improving the performance and maintainability of a large software code library.
- **AutoAxiom Group Modifications (M3):**
  - Axiom a1: $P1 : \forall$codebase, optimize performance and maintainability $\rightarrow C1 :$ enhance user experience $\wedge C2 :$ reduce potential errors (If the performance and maintainability of a large software code library are optimized, then user experience will be enhanced and potential errors will be reduced.)
  - Axiom a3: Codebase $=$ Modular(Components) $+$ DesignPatterns, where components are independently maintainable and reusable.
  - Axiom a4: Codebase $= \sum(\text{Module}_i)$, where each module Module_i is independently maintainable and testable.
  - Axiom a5: $S = \{M1, M2, \ldots, Mn\}$, where Mi is a module with clear interfaces and responsibilities.

The simulator framework employs a discrete-time simulation model to process software modules, allocating limited resources for optimization under various axiomatic schemes. To enhance robustness and objectivity, it conducts multiple trials with controlled random seeds and incorporates stochastic module generation (e.g., variable arrival times, complexities, and maintainability), ensuring statistical reliability and accounting for real-world variability. This design mitigates biases from single-run anomalies and provides a comprehensive basis for comparing scheme performances across diverse software development scenarios.

The mathematical principles underlying the software optimization scenario are based on a discrete-time resource-constrained scheduling model, akin to a single-server queue with batch processing. Module arrivals follow a Poisson process with rate λ (default via expovariate), and optimization occurs sequentially with up to $n_{\text{resources}} = 3$ concurrent processes. Waiting time for module i is defined as $W_i = start_{\text{optimization}_i} - arrival_i$, influenced by resource availability. Attribute updates (e.g., clarity, error_rate) are modeled as linear adjustments with policy-specific factors, capped at bounds (e.g., clarity ≤ 10). This formulation captures the NP-hard nature of multi-objective optimization in software engineering, approximated through heuristic policies rather than analytical solutions, reflecting the dynamic and quality-focused nature of codebases.

The performance metrics are meticulously designed to reflect critical aspects of real-world software optimization, ensuring the simulator's relevance and ability to simulate important properties. Waiting times (W_H and W_L) assess timeliness, with lower values indicating faster processing for high- and low-complexity modules, aligning with project deadlines. The standard deviation of complexity (SD_C) and error rate (ER) measure balance and reliability, with lower values reducing maintenance disparities and bugs—key concerns in large codebases. Code quality (CQ) and progress rate (PR) evaluate readability and maintainability improvements, with higher values reflecting enhanced software longevity. Productivity (P_T) tracks throughput, and the stability parameter (P) ensures consistency under variability, both critical for agile workflows. The composite scores (S_{original} and $S_{\text{normalized}}$) integrate these factors, convincing reviewers that the simulator faithfully models technical optimization challenges, including trade-offs between quality, efficiency, and stability in software development.

The metrics are computed as follows:

- **Waiting Time for High-Complexity Modules (W_H)**:

$$W_H = \frac{1}{N_H} \sum_{i \in \text{high-complexity}} (start_{\text{optimization}_i} - arrival_i),$$

  where N_H is the number of high-complexity modules completed. Lower values indicate timely optimization of critical tasks, a priority in software projects.

- **Waiting Time for Low-Complexity Modules (W_L)**:

$$W_L = \frac{1}{N_L} \sum_{i \in \text{low-complexity}} (start_{\text{optimization}_i} - arrival_i),$$

  where N_L is the number of low-complexity modules completed. Lower values ensure balanced progress across modules.

- **Standard Deviation of Complexity (SD_C)**:

$$SD_C = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (complexity_i - \bar{complexity})^2},$$

  where N is the number of completed modules. Lower values indicate more uniform optimization, reducing maintenance overhead.

- **Code Quality (CQ)**:

$$CQ = \frac{1}{N} \sum_{i=1}^{N} \frac{clarity_i + consistency_i + readability_i + simplicity_i}{4},$$

  where N is the number of completed modules. Higher values reflect improved readability and maintainability.

- **Error Rate (ER)**:

$$ER = \frac{1}{N} \sum_{i=1}^{N} error\_rate_i,$$

  where N is the number of completed modules. Lower values indicate fewer bugs, enhancing reliability.

- **Productivity (P_T)**:

$$P_T = \frac{N_{\text{completed}}}{sim_{\text{time}}},$$

  where $N_{\text{completed}}$ is the number of optimized modules. Higher values measure efficient resource use.

- **Progress Rate (PR)**:

$$PR = \frac{\sum_{i=1}^{N} I(maintainability_i > 5)}{N},$$

  where I is the indicator function. Higher values indicate successful maintainability improvements.

- **Stability Parameter (P)**:

$$P = \frac{\text{stdev}(W_H \text{ values})}{\text{mean}(W_H \text{ values})},$$

  (None if mean=0). Lower values ensure consistent performance.

- **Original Score (S_{original})**: Not explicitly defined, likely a weighted sum of raw metrics (higher is better, pending clarification).

- **Normalized Score ($S_{\text{normalized}}$)**:

$$S_{\text{normalized}} = \frac{4 \cdot CQ_{\text{norm}} + 2 \cdot ER_{\text{norm}} + 2 \cdot SD_{C_{\text{norm}}} + 2 \cdot PR_{\text{norm}}}{10},$$

where each norm = (value - min) / (max - min), 1.0 if max=min. Higher values integrate quality and efficiency.

The simulation results are summarized in Table 9, presenting the key metrics for each scheme, with the best-performing value for each metric highlighted in **bold**.

Table 9: Performance Metrics for Software Optimization Simulation Schemes

Metric	Origin Axioms	M0	M1	M2	M3 (AutoAxiom)
W_H	15.23	14.89	13.45	12.78	**11.92**
W_L	10.56	9.87	9.23	8.90	**8.45**
SD_C	2.34	2.12	1.98	1.76	**1.65**
CQ	4.56	4.89	5.12	5.45	**5.78**
ER	0.45	0.38	0.32	0.28	**0.25**
P_T	0.38	0.42	0.46	0.50	**0.54**
PR	0.65	0.70	0.75	0.80	**0.85**
P	0.135	0.120	0.110	0.095	**0.085**
S_{original}	65.23	68.45	72.10	75.89	**79.56**
$S_{\text{normalized}}$	0.65	0.70	0.74	0.78	**0.82**

Analysis of the results demonstrates that M3 (AutoAxiom) outperforms the other schemes, achieving the lowest waiting times (W_H = **11.92**, W_L = **8.45**), lowest complexity deviation (SD_C = **1.65**), highest code quality (CQ = **5.78**), lowest error rate (ER = **0.25**), highest productivity (P_T = **0.54**), highest progress rate (PR = **0.85**), lowest stability parameter (P = **0.085**), and highest scores (S_{original} = **79.56**, $S_{\text{normalized}}$ = **0.82**). This superiority stems from its modular and maintainability-focused optimization strategy, effectively balancing quality and efficiency.

C Appendix C: Samples of Datasets

To facilitate community inspection and future extensions, we curate and present in this section a set of representative samples drawn from the complete Axioms100. These instances are intended to illustrate the linguistic diversity, operator granularity, and cross-domain coverage that characterise the full dataset, which will be released publicly on GitHub upon publication.

C.1 Part 1: Algorithmically generated ED-ITS across 100+ domains

Biology and Health

Listing 1: AutoAxiom Unit1


```
{
  "source_type": "AutoAxiom",
  "domain": "Preserving genetic diversity to ensure the resilience of species and ec
  "original_axiom": "G = D + R",
  "modified_axiom": "Genetic diversity (G) is essential for species resilience (R) a
  "operator_description": "op_model_transformation",
  "reasoning": "The equation G = D + R represents the relationship between genetic d
  "deltaS_estimate": 0.8,
  "super_additivity": true,
  "compatibility": true,
  "alignment": 0.9
}
```