Towards More Dependable Specifications: An Empirical Study Exploring the Synergy of Traditional and LLM-Based Repair Approaches

Md Rashedul Hasan*, Mohannad Alhanahnah†, Clay Stevens‡, Hamid Bagheri*
*University of Nebraska-Lincoln, Lincoln, NE, USA
†Chalmers/University of Gothenburg, Sweden
‡Iowa State University, Ames, IA, USA
Emails: mhasan6@cse.unl.edu, Mohannad.alhanahnah@chalmers.se, cdsteven@iastate.edu, bagheri@unl.edu

***Abstract*—Declarative specification languages like Alloy are critical for modeling and verifying complex software systems, yet repairing these specifications remains a significant challenge for ensuring software dependability. This study conducts the first comprehensive empirical evaluation comparing traditional systematic repair techniques with emerging Large Language Model (LLM)-based approaches across two established benchmarks, analyzing over 1,900 Alloy specifications. By systematically analyzing repair success rates, ground truth similarity, and repair generation strategies, we reveal nuanced performance characteristics of different repair methodologies. Our findings demonstrate that while traditional tools excel in systematic fault localization and achieving high ground truth similarity, LLM-based techniques—particularly multi-round prompting approaches—offer unique capabilities in addressing complex specification errors, with some hybrid approaches achieving repair rates of up to 85.5%. Critically, we show that integrating traditional fault localization techniques with LLM-based repair strategies can significantly enhance overall repair effectiveness and specification dependability. This research provides a large-scale empirical evaluation of how various Alloy repair techniques work in synergy, offering valuable insights that chart a promising path for future automated specification repair approaches and contribute to the development of more reliable and secure software systems.**

I. Introduction

As our reliance on software systems grows, particularly in safety-critical domains, ensuring their reliability becomes paramount. Declarative specification languages play a crucial role in addressing this challenge by enabling developers to precisely define system behavior without implementation details. This approach facilitates the modeling, verification, and analysis of complex systems, catching subtle yet potentially dangerous bugs at the design level before implementation begins [1]–[3]. Among these, the Alloy specification language [4] stands out for its powerful capabilities in utilizing relational algebra and first-order logic to model systems and verify their properties, providing a formal framework applicable across many domains in software engineering.

Alloy's versatility is evident in its wide-ranging applications, from software verification and security analysis to systematic test case generation [5]. In software verification, Alloy detects errors and inconsistencies early in the development process, ensuring systems behave as intended [6]–[10]. For safety and security analysis of modern platforms, like the Internet of Things (IoT) and Android, Alloy helps identify and mitigate otherwise undetected vulnerabilities [11]–[18]. Additionally, Alloy aids in the systematic generation of test cases, enhancing software robustness through comprehensive testing coverage [19], [20]. The integration of Alloy with the Alloy Analyzer further enhances its functionality by automating the verification of specified properties within given constraints, providing immediate feedback and facilitating a more efficient, iterative approach to specification development. Notably, Alloy has been successfully applied to diverse critical systems, ranging from web security protocols to surgical robots [1], [21], demonstrating its effectiveness in enhancing software reliability and safety across a wide spectrum of domains. This broad applicability underscores Alloy's capability to address complex dependability challenges in various critical software systems.

However, the process of debugging and repairing these specifications remains a significant challenge [22], [23]. Unlike imperative programming languages, where automated program repair (APR) techniques have seen substantial advancements, declarative specification repair presents unique complexities that demand innovative approaches. This gap is particularly critical for Alloy users facing challenges in debugging and correcting subtle bugs in complex system specifications. A recent empirical study analyzing over 93,000 Alloy specifications found that approximately 75% of specifications written by novice users are either incorrect or fail to compile, with 45.3% of the specifications being faulty and an additional 29.28% not compiling, highlighting the urgent need for effective debugging and repair techniques in declarative specification languages [23].

The inherent characteristics of declarative languages—emphasizing *what* a system should do rather than *how* it should be implemented—introduce nuanced challenges for repair techniques. Specifications often involve intricate logical constraints, relational modeling, and complex state representations that traditional repair methods struggle to address comprehensively. Moreover, the absence of explicit

procedural logic means that repair strategies must navigate abstract semantic relationships rather than straightforward syntactic transformations.

In response to these challenges, researchers have developed a range of approaches to Alloy repair [24]–[32]. Traditional tools like ARepair [25], ICEBAR [26], BeAFix [27], and ATR [28] have made initial strides, employing strategies such as test-based repair, counterexample-driven approaches, and bounded exhaustive exploration. These tools represent the first wave of solutions addressing the previously identified lack of APR techniques for declarative specifications.

Recent technological developments, particularly the emergence of Large Language Models (LLMs), have opened new frontiers in automated repair techniques. These models offer unprecedented capabilities in understanding context, generating semantically meaningful repairs, and bridging gaps between human intent and computational implementation. However, their application in declarative specification repair remains largely unexplored, presenting both exciting opportunities and significant methodological challenges. Recent studies have begun to investigate LLM-based techniques, offering new possibilities in natural language understanding and code generation for specification repair [33], [34].

Our study emerges from this complex landscape, seeking to systematically investigate and evaluate approaches to Alloy specification repair. We recognize that no single repair technique can universally address all specification defects. Instead, we aim to develop a nuanced understanding of how different repair methodologies—ranging from traditional systematic approaches to cutting-edge machine learning techniques—can complement and enhance each other.

The research presented here is motivated by three critical observations in the field of specification repair:

1) **Methodological Diversity:** Existing repair techniques employ fundamentally different strategies, from test-driven approaches to counterexample-based methods and emerging LLM-driven techniques.
2) **Limited Comprehensive Evaluation:** Prior studies have typically focused on individual repair techniques, leaving a significant gap in understanding their relative strengths, limitations, and potential synergies.
3) **Technological Convergence:** The rapid advancement of machine learning techniques presents unprecedented opportunities for integrating traditional systematic approaches with intelligent, context-aware repair strategies.

This study represents the first large-scale empirical evaluation of how various Alloy repair techniques work in conjunction. By conducting a comprehensive, empirical investigation into Alloy specification repair techniques, we seek to address challenges in automated specification repair and provide insights to guide future research and development. Our study utilizes established benchmarks, including the ARepair benchmark [35] and the Alloy4Fun benchmark [36], to evaluate the effectiveness of various repair techniques across different problem domains. This research explores key questions about the effectiveness of individual Alloy repair techniques, their complementarity, and the potential for hybrid approaches that leverage the strengths of both traditional and LLM-based methods. We assess repair effectiveness through quantitative measures such as repair success rates and similarity to ground truth, while also considering qualitative aspects of the generated repairs. Through rigorous methodology and analysis, we aim to contribute insights to the field of declarative specification repair, paving the way for more robust, efficient, and intelligent repair strategies.

Our study presents the following significant findings:

- **Finding 1:** Alongside traditional repair techniques, LLMs have shown significant effectiveness in repairing Alloy specifications, particularly using *multi-round* prompting techniques. In the ARepair benchmark, Multi-Round LLM approaches outperformed traditional methods, while in the Alloy4Fun benchmark, they were competitive with top-performing traditional techniques. This suggests that leveraging advanced algorithms and machine learning models can enhance repair performance beyond what traditional methods alone can achieve (Section IV-A, Table I).
- **Finding 2:** Both traditional and LLM-based techniques achieved high similarity to the ground truth in certain scenarios, indicating their potential to produce high-fidelity fixes. Notably, traditional techniques like ATR, ICEBAR, and BeAFix consistently achieved higher similarity scores, especially in the Alloy4Fun benchmark. This highlights the strength of established methods in maintaining the integrity of the original specification (Section IV-A, Figure 2).
- **Finding 3:** Correlation analyses revealed varying patterns of similarity among different repair methods across benchmarks. While traditional techniques showed strong internal correlations, LLM-based approaches demonstrated distinct patterns. This suggests that integrating multiple repair methods, especially across different paradigms, could lead to more robust and comprehensive repair strategies (Section IV-B, Figure 3).
- **Finding 4:** Combining fault localization capabilities of traditional tools with the bug fixing strengths of LLM-based techniques showed potential for improved results, particularly in enhancing repair case ratios. The effectiveness of these combinations varied across benchmarks, with some pairings (e.g., ATR with Multi-Round LLM approaches) showing significant improvements in the Alloy4Fun benchmark. This synergy between approaches suggests a promising direction for more effective and efficient code repair processes (Section IV-C, Figure 4).

To summarize, the novel contribution of this study is a large-scale empirical evaluation comparing traditional and LLM-based Alloy repair methods across two benchmarks, uncovering their complementary strengths. It introduces hybrid repair strategies that combine systematic fault localization with iterative multi-round LLM prompting, significantly improving

repair success rates and advancing our understanding of specification repair methodologies.

The remainder of this paper is structured as follows: Section 2 provides background on Alloy and existing repair techniques. Section 3 details our study design and methodology. Section 4 presents our experimental results and analysis. Section 5 presents potential threats to the validity of our study and the measures taken to mitigate them. Section 6 discusses the implications of our findings and potential future directions. Finally, Section 7 concludes the paper with a summary of our key insights and their significance for the field of automated specification repair.

II. Alloy: A Lightweight Specification for Dependable Systems

Alloy is a lightweight formal specification language and analysis tool designed to enhance software dependability throughout the development lifecycle [37]. It enables developers to model and analyze complex software designs, with a particular focus on critical system properties and dependability cases. Using a simple yet powerful syntax based on first-order logic and relational algebra, Alloy allows for the expression of intricate structural constraints and behaviors of software systems. The tool's primary strength lies in its ability to formally verify system properties and invariants before implementation, thereby significantly improving software reliability and robustness. By facilitating early detection of design flaws, inconsistencies, and potential failure modes, Alloy plays a crucial role in building dependable software systems. It supports the creation and validation of dependability cases, allowing developers to systematically reason about and demonstrate the reliability, safety, and security of their systems.

The Alloy Analyzer is a crucial component of the Alloy toolkit. It automatically analyzes Alloy specifications to find instances (examples) that satisfy the model's constraints or counterexamples that violate specified assertions. The Analyzer translates Alloy models into boolean satisfiability (SAT) problems and uses off-the-shelf SAT solvers to find solutions. This approach allows for bounded verification, where the Analyzer explores all possible scenarios within a specified scope, providing concrete feedback to developers.

To illustrate the key concepts of the Alloy formal specification language and demonstrate how automated repair tools can address subtle bugs, this section presents a simplified file system specification derived from the Alloy4Fun benchmark. This example will showcase Alloy's syntax, basic concepts, and a common bug that developers might encounter.

This specification demonstrates several key Alloy concepts. Signatures, defined on lines 1-12, represent sets of atoms. The 'abstract' keyword on line 1 indicates that Key cannot have direct instances, while 'extends' on line 2 creates a subset relationship. Within signatures, fields define relations, such as 'keys' in Room (line 4) which relate each Room to a set of Keys. Multiplicity keywords like 'lone' in FrontDesk (line 10) specify that each Room maps to at most one RoomKey.

```
1  abstract sig Key {}
2  sig RoomKey extends Key {}
3  sig Room {
4    keys: set Key
5  }
6  sig Guest {
7    keys: set Key
8  }
9  one sig FrontDesk {
10   lastKey:  Room -> lone RoomKey,
11   occupant: Room -> lone Guest
12 }
13
14 fact HotelInvariant {
15   all r: Room | some FrontDesk.lastKey[r]
16   all r: Room, k: Key | k in r.keys iff k = FrontDesk.
     ↪ lastKey[r] or k in FrontDesk.occupant[r].keys
17   all g: Guest, r: Room | g = FrontDesk.occupant[r] iff
     ↪ FrontDesk.lastKey[r] in g.keys
18 }
19
20 pred checkIn(g: Guest, r: Room, k: RoomKey) {
21   r not in FrontDesk.occupant.Guest
22   no g.keys
23   FrontDesk.lastKey'  = FrontDesk.lastKey ++ r->k
24   FrontDesk.occupant' = FrontDesk.occupant + r->g
25   g.keys' = g.keys + k
26   r.keys' = r.keys + k
27 }
```

Fig. 1: Faulty Alloy Specification of Hotel Key Management.

Facts, such as HotelInvariant (lines 14-18), define constraints that always hold in the model. Predicates, like checkIn (lines 20-27), describe operations, using primed variables (e.g., lastKey') to represent post-state values. Alloy also employs quantifiers and operators: 'all' (lines 15-17) is a universal quantifier, while operators like 'in', '->', and '+' manipulate sets and relations.

The subtle bug in this specification lies in the checkIn predicate, specifically on line 22: `no g.keys`. This constraint requires that a guest have no keys when checking in, which is overly restrictive. It prevents scenarios where a guest might legitimately hold keys to other rooms or check into multiple rooms. This bug exemplifies the challenges in formal specification: It requires a deep understanding of the real-world system being modeled, demands a delicate balance in constraint formulation, and might only reveal its full impact when considering diverse use cases.

Automated repair tools can significantly aid in addressing such nuanced issues. These tools might identify the problem where valid check-in scenarios are prevented. For instance, a tool could suggest replacing line 22 with a more precise constraint: `k not in g.keys`. This modification would ensure the guest doesn't already have the specific key being issued, rather than requiring no keys at all. The challenge for automated repair tools in this case is to identify the overly restrictive nature of the constraint that illustrates valid scenarios the current specification doesn't allow.

By addressing such subtle logical errors, automated repair tools can significantly aid developers in creating correct and robust formal specifications, especially when dealing with complex structural constraints like those found in access control systems. While our example focused on an overly

restrictive constraint, automated repair tools can address various types of bugs in Alloy specifications. These may include under-constraint issues (where the model allows unintended behaviors), over-constraint problems (where valid scenarios are inadvertently excluded), logical inconsistencies, and syntax errors. These tools must not only identify bugs but also propose fixes that maintain the overall integrity of the model while allowing for all valid real-world scenarios.

III. Study Design

Automated repair of declarative specifications presents complex challenges that demand sophisticated approaches to identify, localize, and correct software defects. This section outlines our methodological framework for evaluating and understanding the landscape of Alloy specification repair techniques. We present a comprehensive overview of selected repair tools and techniques, describe the benchmark specifications used for evaluation, detail the metrics employed to assess repair effectiveness and provide the experimental settings and implementation details.

A. Research Questions

In this study, we address three fundamental research questions that explore the landscape of Alloy specification repair:

RQ1: (Effectiveness) How effective are individual Alloy repair techniques—including traditional tools and LLM-based approaches?

This question systematically assesses the performance of diverse repair methodologies across established benchmarks. By evaluating each technique's ability to generate accurate repairs, we aim to understand the strengths and limitations of current approaches in addressing specification defects. Our investigation will provide insights into the effectiveness of both traditional systematic repair tools and emerging large language model-based techniques.

RQ2: (Complementarity) To what extent do these techniques produce similar or complementary repair outcomes?

Our investigation explores the potential synergies and variations among different repair approaches. We aim to determine whether techniques produce similar repairs or offer unique solutions to specification challenges. By analyzing the correlation and divergence between repair methods, we seek to understand how different approaches might complement each other in addressing complex specification defects.

RQ3: (Combination) What is the potential for hybrid approaches that leverage strengths from both traditional methods and LLM-based techniques?

Building upon insights from the previous research questions, this inquiry explores the possibility of creating more robust repair strategies by integrating different methodological approaches. We investigate how the systematic fault localization capabilities of traditional tools might be combined with the flexible, context-understanding strengths of large language models to develop more comprehensive repair techniques.

To address these research questions, we designed a comprehensive study that systematically examines various Alloy specification repair techniques.

B. Selected Tools and Techniques

In this study, we conduct a comparative analysis that evaluates the performance of four state-of-the-art traditional Alloy repair techniques against several recent methods that utilize Large Language Models for conducting repairs. These tools and techniques serve as a baseline for several studies that have achieved remarkable results in fixing faulty Alloy specifications [26]–[28], [33]–[35]. The specific tools used in our study are described in detail below.

1) Traditional Alloy Repair: We focus on four traditional repair tools that do not use LLMs in their process. ARepair [25], ICEBAR [26], BeAFix [27], and ATR [28]. Each tool uses a distinct approach to specification repair.

a) Arepair [25]: The ARepair tool is designed to address faulty Alloy models by taking them as input alongside a suite of tests that outline the desired model properties. It generates a corrected model that satisfies all provided tests. Utilizing tests crafted within the AUnit framework—an established method for unit testing Alloy models—ARepair identifies and rectifies faults in the models. ARepair operates by accepting a defective Alloy model responsible for triggering failed tests, along with an AUnit test suite. ARepair explores the solution space iteratively, aiming to rectify the model so that all tests pass successfully. This iterative process may involve multiple modifications to the model, with ARepair employing a greedy approach until it either discovers a solution that resolves all test failures or exhausts all possibilities without success. ARepair's main limitation is its reliance on user-provided tests, which can cause overfitting—producing repairs that pass the tests but may not meet the intended specification.

b) ICEBAR [26]: ICEBAR is an iterative, counterexample-driven process designed to generate and validate repairs for flawed Alloy specifications. The process begins by accepting a defective Alloy specification along with a failing property-based oracle as inputs. Utilizing Alloy's counterexamples, ICEBAR constructs tests that are processed by ARepair (as previously introduced) to produce a candidate repair. This candidate is then evaluated against the property oracle to check for overfitting: if it satisfies the oracle, it is considered a successful repair; if not, ICEBAR generates additional counterexamples to further refine the test suite. This iterative cycle continues until either a successful repair is identified or no viable repairs can be found. ICEBAR employs various mechanisms to generate counterexamples from failing predicates and assertions, each varying in reliability. Overall, ICEBAR's refinement strategy enhances robustness, but its success is limited by the quality of generated counterexamples.

c) BeAFix [27]: BeAFix is an automated repair technique specifically developed for addressing faults in Alloy models. It employs a distinct strategy that utilizes bounded exhaustive exploration to navigate the space of potential repair candidates, significantly enhanced by a robust pruning method that improves scalability. Unlike other Alloy repair methods, such as ARepair [25] and ICEBAR [26], which depend on unit tests for validation, BeAFix operates without them, instead

leveraging assertions inherent to the Alloy specification. This allows for a more integrated approach to model correction. The process begins with BeAFix mutating a faulty Alloy model to generate repair candidates, diligently exploring the state space while applying pruning strategies to optimize efficiency and manage the rapid growth of potential solutions. While BeAFix can utilize tests as oracles for validation, it primarily relies on property-based oracles found in Alloy models through predicate satisfiability and assertion validity checks. BEAFIX's main limitation is the rapid growth of the repair search space as the number of suspicious locations and allowed mutations increases, which can significantly impact efficiency despite its pruning techniques.

d) ATR [28]: ATR is an automated program repair technique specifically designed for declarative specifications in the Alloy language. It accepts a faulty Alloy specification characterized by violated assertions and produces a repaired specification that satisfies these assertions. The method is grounded in two fundamental principles: first, it analyzes the differences between counterexamples and satisfying instances to inform the repair process; second, it generates repair candidates from predefined templates while systematically pruning the candidate space using both counterexamples and satisfying instances. To identify satisfying instances similar to those produced by the Alloy Analyzer, ATR employs a PMAXSAT solver. Additionally, it leverages an analysis of both satisfying instances and counterexamples to refine its candidate repair templates, enhancing the overall effectiveness of the repair process. Overall, ATR's systematic, template-based approach yields high-quality repairs, but its performance is constrained by the expressiveness and coverage of its predefined templates.

2) LLM based Alloy Repair Techniques: In contrast to those techniques, some recent approaches utilized LLMs—specifically GPT [38]—to propose repairs. Two recently proposed approaches employ versatile prompt engineering techniques to mend flawed Alloy specifications.

a) Single-Round Approach [33]: Hasan et al. [33] leveraged OpenAI's ChatGPT to automate the repair of faulty Alloy specifications using a single, zero-shot prompt. This approach involves providing the model with the faulty specification along with various combinations of critical information:

(a) *Bug location (Loc)*: The approximate location of the bug.
(b) *Fix comment (Fix)*: A description of a potential fix.
(c) *Passing assertion requirement (Pass)*: An assertion that the fix must satisfy.

The study explored five distinct prompt configurations: *Loc+Fix*, *Loc*, *Pass*, *None* (no additional hints), and *Loc+Pass*, each offering insights into the effectiveness of different informational cues in automated specification repair.

b) Multi-Round Approach [34]: Alhanahnah et al. [34] advanced the field by introducing a sophisticated *Multi-Round* approach to LLM prompting, contrasting with prior non-iterative, single-prompt methods. This technique employs a dual-agent LLM framework (Repair Agent and Prompt Agent), leveraging the more advanced GPT-4 model.

The core of this approach lies in its iterative nature; the agents engage in a dialogue with the LLM, refining repairs based on feedback from the Alloy Analyzer. This process creates a dynamic, self-improving repair cycle. Our study replicates their methodology, exploring three feedback levels:

(a) *No-feedback*: A minimalist approach where the Alloy Analyzer provides only a binary fix confirmation, challenging the LLM's autonomous error identification and correction capabilities.
(b) *Generic-feedback*: Simulating a developer's query on a Q&A platform, this setting offers a template-based summary of counterexamples, instances, and errors.
(c) *Auto-feedback*: The most advanced setting, where the Prompt Agent, another LLM, generates tailored feedback based on the Alloy Analyzer's report and proposed specifications, guiding the Repair Agent in a sophisticated AI-to-AI interaction.

C. Benchmarks

For this study, we utilized two distinct benchmark suites: ARepair [35] and Alloy4Fun [36]. These suites have undergone extensive scrutiny and were developed by separate research entities, allowing for equitable comparisons across various methodologies.

The **Alloy4Fun benchmark** [36] consists of *1,936 manually crafted flawed specifications* sourced from submissions across six distinct Alloy problem domains. These domains include modeling challenges such as labeled transition systems (LTS), automated production lines, class registrations, work and source distribution dilemmas, assorted graph properties (including acyclic and completeness conditions), and the conceptualization of a file system trash can. Each specification is designed to encapsulate realistic bugs that Alloy developers might encounter, providing a rich dataset for evaluating repair techniques. This benchmark is particularly valuable because it contains real-world examples of common mistakes and challenges faced by Alloy developers.

In contrast, the **ARepair benchmark** [35] encompasses *38 flawed specifications* derived from a range of 12 Alloy problems. Six of these issues—labeled addr, cd, ctree, farmer, bempl, and others—were sourced from the Alloy Analyzer. The remaining flawed specifications were extracted from graduate student assignments, encompassing problems like arr, balancedBST, dll, fsm, and student. This benchmark is particularly valuable as it includes both simple and complex bugs, allowing for a comprehensive assessment of repair tools.

The benchmark datasets comprise specifications ranging from tens to hundreds of lines, each harboring genuine bugs crafted by human developers. The issues encapsulated within the benchmarks vary widely: they range from simple faults amendable by adjusting a single operator to intricate defects necessitating the synthesis of new expressions or the substitution of entire predicate bodies. Both benchmarks include correct specification versions that serve as reference points for validating result accuracy.

D. Metrics

This study employs three metrics to evaluate the effectiveness of the repair techniques examined. The first metric indicates whether a given technique successfully repaired each flawed specification, while the remaining two metrics assess the token based similarity and the syntactic similarity of each repair candidate—successful or otherwise—to the ground truth for each benchmark specification. For each repair technique, the findings present the arithmetic mean of each metric across the relevant benchmark(s).

a) Repair (REP): The first metric, Repair (REP), indicates whether a given repair technique can successfully fix a flawed specification. It is computed using the Alloy Analyzer to run each command in both the proposed fix and its corresponding ground truth. For each command in the ground truth specification, results are compared with those from the proposed fix. If any results differ, a REP of 0 is assigned to indicate an unsuccessful repair. Conversely, if all results match, a REP of 1 is assigned to signify a successful repair.

b) Token Match (TM): Token Match (TM) represents the token-level similarity of the text in each proposed repair compared to the text of the ground truth for the corresponding specification. It is computed by calculating the sentence-level BLEU score [39] for the tokens in each file, with tokens separated based on whitespace. The score ranges from 0 (no tokens match) to 1 (the files are effectively identical).

c) Syntax Match (SM): Syntax Match (SM) indicates the similarity of the parse trees of each file according to the syntax of the Alloy language. This measure ignores whitespace and other differences irrelevant to the Alloy Analyzer, indicating how structurally similar each fix is to the ground truth. SM is computed by constructing a parse tree using a custom parser built from an ANTLR grammar adapted from Eid [40]. The parse trees are then compared by computing subtree kernel similarity [41]–[43]. SM can range between 0 and 1; a score of 0 indicates that none of the subtrees from the ground truth appear in the parse tree of the fix, while a score of 1 indicates that both parse trees are structurally identical.

E. Experimental Settings

To evaluate the repaired specifications, we utilized Alloy Analyzer 4.2 to compare the repair results against the ground truth. For the traditional techniques, the repaired files provided by the authors of each tool in their prior work [25]–[28] were verified using the Alloy Analyzer to compute the relevant metrics. To compute the Repair (REP) metric, we implemented a custom Java program that executes the Alloy Analyzer via its Java API. This program runs each command in both the proposed fixes and their corresponding ground truths, outputting a JSON file summarizing the results.

For testing the LLM-based techniques, we developed a program using Python version 3.11 to leverage API services from OpenAI and Microsoft Azure for GPT-4, which is the most recent version of the GPT model available at the time of this writing. Additionally, a specialized parser was developed to address challenges posed by unique scenarios that could hinder the extraction of proposed specifications.

The experiments were conducted on a system equipped with a 3.5 GHz Quad-Core Intel Core i7 processor and 32 GB of RAM, running macOS Monterey. The system was configured with Oracle Java SE Development Kit 8u202 (64-bit).

IV. Experimental Results

A. Results for RQ1: Effectiveness of Existing Techniques

Repair Performance. Table I presents a comprehensive comparison of the REP scores (number of specifications repaired) for all tested Alloy repair techniques, including four state-of-the-art traditional repair tools—ATR, ARepair, BeAFix, and ICEBAR—and two LLM-driven techniques under different settings and using different underlying models.

For the smaller ARepair benchmark of 38 specifications, the Multi-Round LLM approach fared particularly well, scoring three of the top five highest REP scores across all techniques. The Multi-Round approach using generic feedback outperformed all other techniques by repairing 29 specifications. BeAFix scored highest among the traditional techniques with 24 repairs, closely followed by ATR (22 repairs) and ICEBAR (21 repairs). ARepair itself repaired only 9 specifications in its own benchmark. The Single-Round LLM approach produced mixed results. The prompt which included the bug location performed well (21 repairs), but the prompt requiring passing the assertions performed worst overall, only repairing 4 of 38 specifications. In the Alloy4Fun benchmark consisting of 1936 specifications, the Multi-Round LLM approach again performed well on all settings, scoring three of the top four highest REP scores. ATR repaired the most specifications of the traditional approaches and third most overall with 1286 repairs. The Single-Round LLM approach performed relatively poorly across all prompt settings, surpassing only ARepair. ARepair performed the worst overall on the Alloy4Fun specifications, only repairing 185. Overall, the Multi-Round approach and ATR outperformed all other techniques, each repairing more than 1300 of the 1974 specifications in total.

> The results indicate that the Multi-Round LLM approach significantly outperformed all other techniques, repairing 76.3% (29 out of 38) specifications in the ARepair benchmark and 69.6% (1348 out of 1936) specifications in the Alloy4Fun benchmark. Traditional tools also demonstrated strong performance, with ATR achieving 66.4% (1286 out of 1936) repairs in the Alloy4Fun benchmark. In contrast, ARepair itself performed the weakest, repairing only 23.7% (9 out of 38) of its own benchmark, highlighting the potential of LLM-based methods in Alloy repair tasks.

B. Results for RQ2: Complementarity and Synergies in Repair Techniques

This section addresses our second research question (RQ2), which explores the complementarity and potential synergies

TABLE I: Comparing REP scores for state-of-the-art traditional Alloy repair techniques (ARepair [35], ICEBAR [26], BeAFix [27], ATR [28]) and LLM-based Alloy repair studies (Single-Round [33], Multi-Round [34]) on the ARepair and A4F benchmarks, using various combinations of settings.

	Model	Total # spec	Traditional				Single-Round (Prompt Settings)					Multi-Round (Feedback Settings)		
			ARepair	ICEBAR	BeAFix	ATR	Loc+Fix	Loc	Pass	None	Loc+Pass	None	Generic	Auto
A4F Benchmark	classroom	999	88	424	387	688	139	231	94	88	162	667	593	553
	cv	138	2	86	82	38	58	50	43	04	53	119	117	117
	graphs	283	19	237	232	260	78	109	90	20	75	158	167	180
	lts	249	1	73	41	70	91	70	49	21	53	51	51	51
	production	61	27	36	56	43	28	32	24	12	26	161	170	158
	trash	206	48	195	183	187	7	5	3	2	5	192	192	178
	Summary	**1936**	**185**	**1051**	**981**	**1286**	**401**	**497**	**303**	**147**	**374**	**1348**	**1290**	**1237**
ARepair Benchmark	addr	1	1	1	1	1	1	0	0	0	1	0	1	0
	arr	2	2	2	2	1	1	1	1	0	1	0	1	0
	balancedBSt	3	1	2	1	1	3	2	2	0	0	1	2	0
	bempl	1	0	1	0	1	0	1	1	1	1	0	0	1
	cd	2	0	2	2	2	1	1	2	0	2	2	2	2
	ctree	1	1	0	0	0	0	0	1	0	1	1	1	1
	dll	4	0	3	3	2	4	4	3	0	1	4	4	4
	farmer	1	0	0	0	0	1	0	0	0	0	0	1	1
	fsm	2	2	2	1	2	2	1	0	0	0	1	1	2
	grade	1	0	1	0	1	1	0	1	0	0	0	0	0
	other	1	0	0	1	1	1	0	0	1	0	1	1	1
	Student	19	2	7	13	10	14	10	15	2	4	14	15	15
	Summary	**38**	**9**	**21**	**24**	**22**	**29**	**20**	**26**	**4**	**11**	**24**	**29**	**27**
	Total	**1974**	**194**	**1072**	**1005**	**1308**	**430**	**517**	**329**	**151**	**385**	**1372**	**1319**	**1264**

among different repair techniques. We aim to understand the extent to which these approaches produce similar or unique repair outcomes, and how they might complement each other in addressing complex specification defects.

Our analysis focuses on two key aspects: the similarity of repairs to ground truth and the correlation between different repair techniques. By examining these factors, we can gain insights into the strengths and potential complementary nature of various repair approaches.

a) Similarity Analysis: Figure 2 provides a comprehensive overview of the similarity between repairs generated by various techniques and the ground truth—a version of each specification known to be correct. We quantify this similarity using two metrics: Token Match (TM) and Syntax Match (SM), as detailed in Section III-D.

Traditional repair techniques, including ATR, BeAFix, ICEBAR, and ARepair, consistently demonstrate higher similarity scores across all metrics compared to LLM-based techniques (single- and multi-round approaches). ATR achieves the highest scores with a Token Match of 0.985 and a Syntax Match of 0.997, while the best-performing LLM-based technique (Multi-Round_Generic) reaches a Token Match of 0.938 and a Syntax Match of 0.943. SM scores consistently exceed TM scores across most techniques, indicating that structural similarity is better preserved than token-level similarity.

These results suggest that traditional techniques produce repairs that more closely resemble the ground truth in terms of both token and syntactic structure. This high structural fidelity could be attributed to these techniques relying on established patterns and heuristics specific to Alloy specifications. In

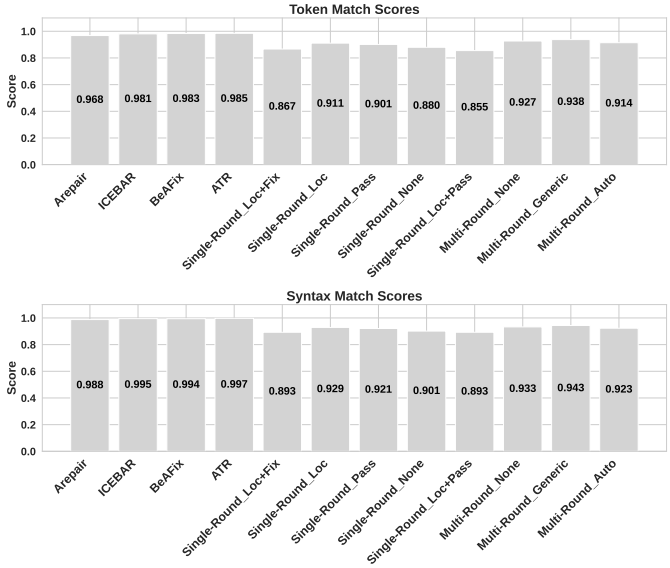

Fig. 2: The bar charts show Token Match (TM) and Syntax Match (SM) similarity scores for 12 specification repair techniques, revealing that traditional tools (ARepair, ICEBAR, BeAFix, ATR) consistently outperform LLM-based approaches in both structural and token-level similarity to the ground truth.

contrast, LLM-based techniques show lower similarity scores, which might indicate more diverse repair outputs. This diversity could result from the inherent creativity and adaptability of language models, potentially leading to solutions that deviate

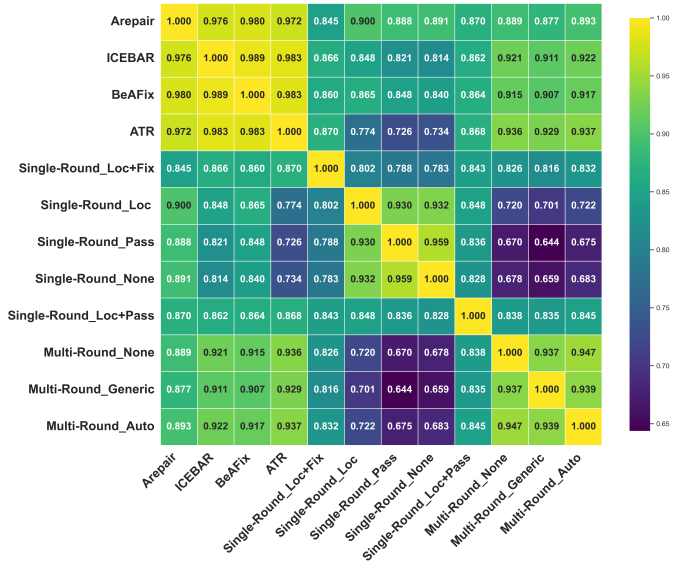

Fig. 3: Correlation analysis of repair techniques. The heatmap displays Pearson correlation coefficients between pairs of repair techniques across two benchmarks. Darker colors indicate stronger correlations. Traditional techniques (ATR, ICEBAR, ARepair, BeAFix) show high internal correlation, while LLM-based techniques (Single-Round and Multi-Round) exhibit distinct patterns, suggesting potential complementarity between different approach types.

more from the exact structure of the ground truth but might address the underlying issues in novel ways.

While higher similarity scores indicate structural fidelity, they do not necessarily reflect comprehensive effectiveness in addressing all faulty specifications. Traditional repair tools may rely on established patterns that enhance similarity but could limit their adaptability to diverse problem contexts compared to LLM-based approaches. The lower scores of LLM-based techniques might suggest a trade-off between strict adherence to known patterns and the ability to generate more flexible, context-aware repairs.

Among the LLM-based approaches, multi-round techniques generally outperform single-round ones, with Multi-Round_Generic showing the highest scores. This suggests that iterative refinement can lead to improvements in the final repair's similarity to the ground truth.

b) Correlation Analysis: To further explore the potential for complementarity among current Alloy Repair techniques, we calculated the Pearson correlation [44] for each pair of techniques. This analysis was based on the match metrics introduced in Section III-D. The Pearson correlation is a widely employed measure for gauging the level of correlation between two data sets [45], [46]. A high Pearson correlation coefficient indicates that the two repair techniques generate similar repairs for each specification. Conversely, a lower Pearson value suggests a lack of correlation between their repairs, potentially indicating opportunities for complementarity.

Figure 3 presents a comparative analysis of repair technique correlations across the two benchmarks, revealing distinct patterns in the relationships between different repair approaches. All correlations reported were found to be statistically significant ($p < 0.001$). The analysis reveals a structure in correlation strengths, with several notable clusters emerging. The strongest correlation is observed between traditional repair techniques with ICEBAR and ATR ($r = 0.983$) demonstrating particularly strong alignment. The traditional repair techniques form a tightly correlated cluster with coefficients consistently above 0.972.

A clear demarcation exists between LLM-based specification repair approaches (including both Multi-Round and Single-Round approaches). Single-Round repair techniques demonstrate notably lower correlations with both traditional and Multi-Round approaches, with coefficients as low as 0.644. However, multi-round approaches maintain stronger internal correlations, particularly between Multi-Round_Generic and Multi-Round_Auto ($r = 0.949$), indicating consistency within the Multi-Round paradigm despite their divergence from Single-Round methods.

The consistent high internal correlations among approaches within each type (traditional, single-round, multi-round) indicate a fundamental alignment in their underlying repair mechanisms, potentially suggesting that combining approaches from the same type may not be optimal. At the same time, significant demarcations emerge between approaches of different types, suggesting the promise of combining these techniques to achieve higher repair rates.

The low correlation between single-round and multi-round approaches stems from inherent differences in their repair processes. Consequently, multi-round techniques do not simply extend single-round outputs but instead iteratively explore different repair directions based on feedback mechanisms. This iterative adaptation causes multi-round repairs to diverge from their single-round counterparts, leading to the observed low correlation despite their shared starting point.

The effectiveness of Multi-Round techniques across benchmarks highlights their versatility in program repair. This suggests that Multi-Round techniques could serve as a robust foundation for hybrid repair approaches, potentially combined with traditional repair strategies to maximize repair effectiveness across different contexts and bug patterns.

The correlation analysis highlights significant differences between repair technique clusters, suggesting promising opportunities for complementary approaches that combine traditional and LLM-based methods to maximize repair effectiveness across different specification contexts. These findings underscore the importance of considering both structural fidelity and adaptive problem-solving capabilities when developing and selecting specification repair techniques.

TABLE II: Overview of the repair capabilities of hybrid approaches combining traditional and LLM-based techniques.

Traditional Technique	Traditional Technique Repairs	LLM Technique	LLM Technique Repairs	Overlaps	Total Repairs (Unique)
ARepair	194	Single-Round_Loc+Fix	430	32	592
		Single-Round_Loc	517	62	649
		Single-Round_Pass	329	35	488
		Single-Round_None	151	21	324
		Single-Round_Loc+Pass	385	27	552
		Multi-Round_None	1372	142	1424
		Multi-Round_Generic	1319	137	1376
		Multi-Round_Auto	1264	122	1336
ICEBAR	1072	Single-Round_Loc+Fix	430	255	1247
		Single-Round_Loc	517	322	1267
		Single-Round_Pass	329	219	1182
		Single-Round_None	151	98	1125
		Single-Round_Loc+Pass	385	230	1227
		Multi-Round_None	1372	807	1637
		Multi-Round_Generic	1319	788	1603
		Multi-Round_Auto	1264	746	1590
BeAFix	1005	Single-Round_Loc+Fix	430	259	1176
		Single-Round_Loc	517	314	1208
		Single-Round_Pass	329	219	1115
		Single-Round_None	151	98	1058
		Single-Round_Loc+Pass	385	227	1163
		Multi-Round_None	1372	768	1609
		Multi-Round_Generic	1319	742	1582
		Multi-Round_Auto	1264	697	1572
ATR	1308	Single-Round_Loc+Fix	430	296	1442
		Single-Round_Loc	517	385	1440
		Single-Round_Pass	329	250	1387
		Single-Round_None	151	127	1332
		Single-Round_Loc+Pass	385	109	1584
		Multi-Round_None	1372	1003	1677
		Multi-Round_Generic	1319	970	1657
		Multi-Round_Auto	1264	913	1659

C. Results for RQ3: Hybrid Approaches: Leveraging Traditional and LLM-based Techniques

Building upon the insights gained from our analysis of individual repair techniques and their correlations, we now turn our attention to RQ3: What is the potential for hybrid approaches that leverage strengths from both traditional methods and LLM-based techniques? This section explores the synergistic possibilities of combining these diverse repair paradigms to enhance overall repair effectiveness.

Our investigation into hybrid approaches was motivated by the complementarity observed between traditional and LLM-based techniques, as highlighted in the correlation analysis from RQ2. To assess the potential of these hybrid strategies, we systematically combined each of the four traditional approaches with eight LLM-based approaches, encompassing both single-round and multi-round variants.

Figure 4 presents a comprehensive view of these combinations through 32 Venn diagrams, each representing the repair capabilities of a traditional approach paired with an LLM-based technique. This visualization allows for a clear comparison of the unique and overlapping repair capabilities of each hybrid combination. Table II provides an overview of the repair capabilities of hybrid approaches that combine traditional and LLM-based techniques.

Our analysis reveals that hybrid approaches consistently outperform their constituent individual techniques. The most striking improvements were observed when combining traditional methods with multi-round LLM-based approaches. For instance, the combination of ATR with the Multi-Round_None LLM approach achieved the highest overall performance, successfully repairing 1,677 out of 1,974 faulty specifications (85.5%). This represents a substantial improvement over ATR's individual performance of 1,308 repairs (66.3%), demonstrating the significant potential of hybrid strategies.

The effectiveness of hybrid approaches varied across different traditional techniques. ICEBAR, when combined with Multi-Round_None, showed a remarkable increase in repair capability, addressing 1,637 faulty specifications (82.9%) compared to its individual performance of 1,072 repairs (54.3%). Similarly, BeAFix paired with Multi-Round_None repaired 1,609 specifications (81.5%), a significant jump from its standalone performance of 1,005 repairs (50.9%). Notably, ARepair, which had the lowest individual performance among traditional techniques (194 repairs, 9.8%), showed the most dramatic improvement when combined with LLM-based approaches. Its best hybrid combination, ARepair with Multi-Round_None, successfully repaired 1,424 specifications (72.1%), representing an extraordinary increase in repair capability.

The extent of overlap between traditional and LLM-based repair techniques reveals their complementarity: strong traditional tools like ATR show substantial shared repair space with multi-round LLMs (e.g., 76.7% overlap), while weaker tools like ARepair have much lower overlap, indicating they benefit more from hybridization. Single-round LLMs consistently have lower overlap with traditional methods than multi-round LLMs, and certain pairings—such as ATR with Single-Round_Loc+Pass—achieve minimal overlap but maximize unique repairs, highlighting the strategic value of combining diverse approaches. The consistent superiority of hybrid combinations involving multi-round LLM approaches aligns with our earlier findings from RQ2, where multi-round techniques demonstrated higher similarity to ground truth repairs. This suggests that the iterative refinement process in multi-round approaches is particularly effective when combined with the structured guidance of traditional techniques.

These results underscore the complementary nature of traditional and LLM-based repair techniques. Traditional approaches, with their structured, rule-based strategies, excel at repairing certain types of specification faults. LLM-based techniques, particularly multi-round, contribute adaptability and the ability to generate novel solutions. When combined, these approaches can cover a broader spectrum of specification defects, leading to significantly higher repair rates.

Our analysis of hybrid approaches highlights a promising strategy for enhancing Alloy specification repair by combining traditional and LLM-based techniques, with multi-round LLMs showing the greatest synergistic potential. Evaluating 32 combinations across four traditional and eight LLM-based methods, we observe performance improvements of over 60%, underscoring the complementary strengths of these paradigms. These findings offer a roadmap for developing more robust, adaptive repair tools that leverage both rule-based structure and flexible generative capabilities.

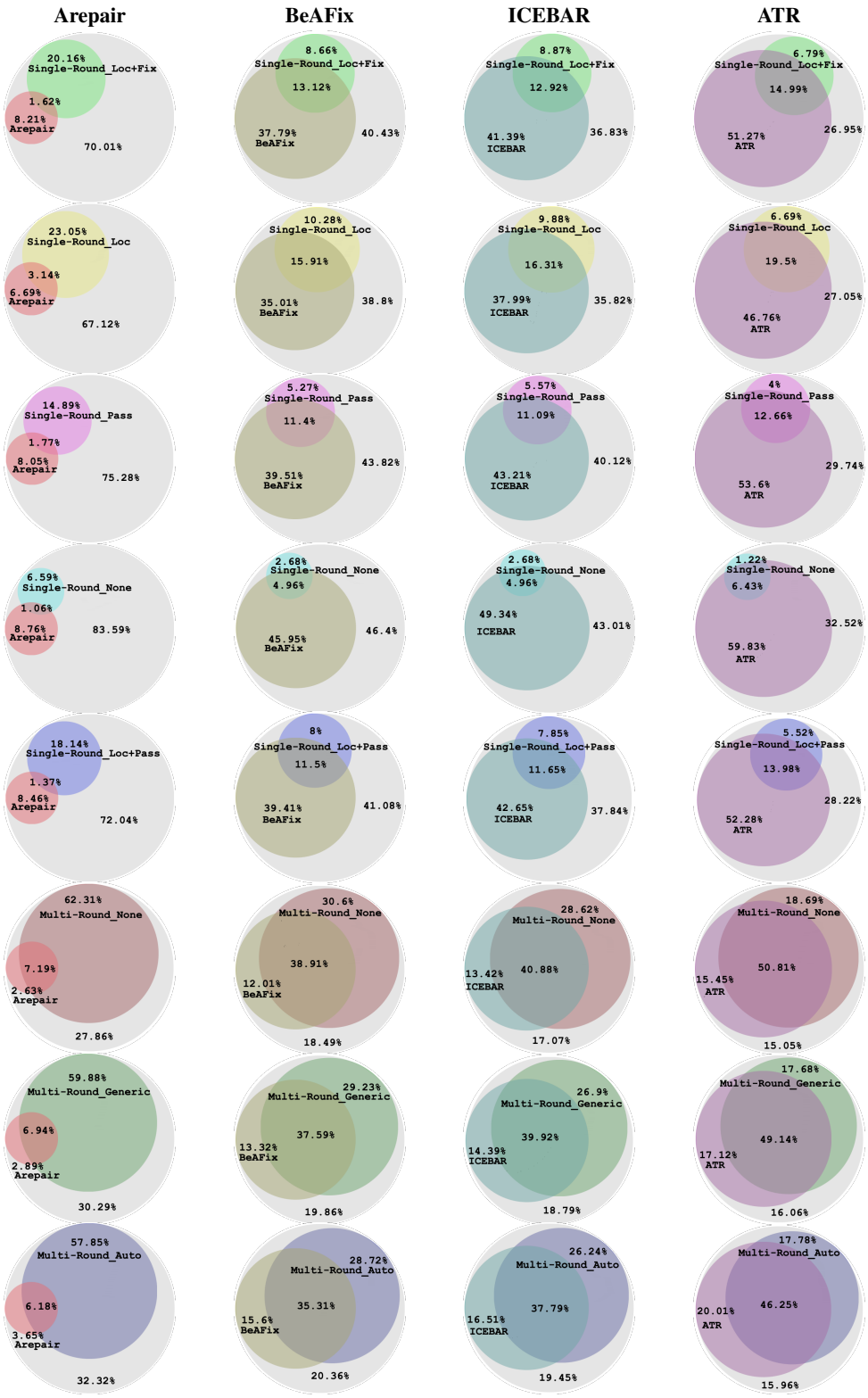

Fig. 4: Venn diagrams illustrating the repair capabilities of hybrid approaches combining traditional and LLM-based techniques for 1,974 Alloy specifications. The figure consists of 32 Venn diagrams arranged in a matrix format. Each column represents one of the four traditional approaches (ARepair, ICEBAR, BeAFix, and ATR), while each row corresponds to a specific LLM-based repair approach. The overlapping regions in each Venn diagram depict the shared repair capabilities, while the non-overlapping areas show the unique contributions of each technique in the hybrid approach. This comprehensive visualization allows for a direct comparison of the synergistic effects across different combinations of traditional and LLM-based repair methods.

V. Threats to Validity

In this section, we discuss potential threats to the validity of our study and the measures taken to mitigate them.

a) Internal Validity Threats: Internal validity concerns the causal relationships identified in our study. One potential threat is the implementation of the repair techniques. To mitigate this, we carefully followed the methodologies described in previous studies. Another threat is the potential bias in the selection of prompts for the Single-Round and Multi-Round LLM approaches. We attempted to address this by using a variety of prompt configurations based on previous research, but there may be other prompt designs that could yield different results.

b) External Validity Threats: The main threats to our external validity are (a) the comprehensiveness of our benchmarks and (b) our focus on techniques which operate on Alloy specifications. Our benchmarks are widely used in research and contain 1,974 faulty specifications; however, they may not include all possible bugs or all Alloy features used in "real-world" Alloy models. The performance of the approaches may vary on other datasets. Alloy was chosen as the specification language due to the breadth of specification repair techniques available for Alloy, which is greater than that for other specification languages such as TLA+. The approaches examined, however, use state-of-the-art techniques (e.g., template-based fault localization, agent-based problem solving) that could be employed in other languages.

c) Construct Validity Threats: To ensure validity of our construct, we selected metrics used in prior studies and calculated those metrics against the provided ground truth (correct) specifications included in each benchmark dataset. REP uses the Alloy Analyzer to compute equisatisfiability between each proposed repair and the ground truth, while TM and SM are computed using similarity analyses proposed in other studies [39], [41]–[43].

VI. Discussion

Our analysis of Alloy specification repair techniques reveals insights into the integration of traditional and LLM-based approaches, highlighting their complementary strengths and the potential for hybrid strategies to enhance repair effectiveness.

Complementarity of Traditional and LLM-based Approaches. Differences in performance can be attributed to both benchmark characteristics and tool design, which differs significantly between the traditional and LLM-based techniques. Our findings illustrate a clear complementarity between traditional repair tools and LLM-based techniques. Traditional repair techniques excel in scenarios involving systematic constraint manipulation and well-defined structural modifications. They demonstrated robust performance on specifications from categories like *classroom*, *graphs*, and *trash* in the Alloy4Fun benchmark, as well as on *arr* and *addr* specifications in the ARepair benchmark.

In contrast, LLM-based approaches, particularly those employing multi-round techniques, exhibit unique capabilities in addressing complex repairs. Their success in modules such as *farmer* and *ctree* underscores their ability to understand implicit constraints and complex state transitions—areas where traditional tools often struggle. For instance, the LLMs effectively repaired the *farmer* module, which requires nuanced reasoning about state transitions.

This complementarity suggests that the future of specification repair lies in integrating these diverse methodologies. By combining the structured, rule-based strategies of traditional tools with the adaptive capabilities of LLMs, we can develop more comprehensive repair systems. For example, ARepair's improvement in RQ3 likely arises because its original repair techniques, though effective at locating faults, struggle to generate accurate fixes. When combined with multi-round LLM, which iteratively refines repairs based on feedback, the hybrid approach leverages ARepair's localization strength and the LLM's synthesis capabilities, significantly enhancing overall repair effectiveness.

The Promise of Hybrid Approaches. Our investigation into hybrid approaches demonstrates substantial potential for enhancing overall repair effectiveness. The superior performance of hybrid strategies, particularly those involving multi-round LLM techniques, indicates a promising direction for future research. The combination of ATR with the Multi-Round_None LLM approach achieved an impressive repair rate of 85.5% across benchmarks, significantly improving upon individual performances.

We chose GPT-4 for its state-of-the-art performance and demonstrated effectiveness in complex reasoning tasks, making it well-suited for evaluating advanced feedback mechanisms in Alloy specification repair. In our study, GPT-4 was deployed in two feedback modes: in Generic-feedback, it receives a templated Alloy Analyzer report summarizing counterexamples and errors, while in Auto-feedback, a dual-agent pipeline uses both the report and the proposed specifications to generate targeted, error-specific instructions for repair. This flexibility allowed us to systematically study how feedback settings affect repair performance. Unlike prior work comparing GPT-3.5 and GPT-4 [33], [34], our study focuses on contrasting GPT-4 with traditional repair tools to isolate the impact of these novel feedback mechanisms and establish a clear baseline for LLM-based automated software repair.

Future repair tools could implement a dynamic approach that selects the most suitable combination of techniques based on the characteristics of faulty specifications. This could involve initial analysis using traditional tools to identify structural issues, followed by LLM-based analysis for semantic understanding and innovative solution generation. Iterative refinement through multi-round approaches could further enhance repair quality.

Implications for Formal Methods and Software Verification. The implications extend beyond Alloy to the broader field of formal methods and software verification. By improving our understanding of effective repair strategies for declarative specifications, we contribute to developing more robust software systems—particularly in critical domains where formal verification is essential.

Furthermore, LLM-based approaches stand to benefit significantly from targeted improvements such as fine-tuning on domain-specific datasets. Fine-tuning these models with a corpus of formal specifications and their corresponding repairs could enhance their understanding of the precise semantics and structural nuances inherent in formal languages. Additionally, iterative feedback mechanisms, reinforcement learning from domain experts, and advanced prompt engineering tailored for formal methods could further refine their output quality. Such enhancements are expected to reduce repair variability, increase consistency, and better align generated repairs with ground truth, thereby amplifying the impact of LLM-based techniques in hybrid repair frameworks.

The success of hybrid approaches signifies a potential paradigm shift in specification repair. This integration may lead to more comprehensive verification processes capable of addressing a wider range of specification errors while reducing time and effort in development phases. Moreover, it could enhance accessibility to formal methods for developers, promoting their adoption in mainstream software development.

Challenges and Future Directions. Despite promising results, key challenges remain: scalability, interpretability, and consistency. Integrating retrieval-augmented generation and leveraging our dataset can enhance fine-tuning, while iterative refinement and continuous learning will improve handling of complex specifications. Addressing scalability is crucial as specifications grow, interpretability remains a barrier due to LLMs' black-box nature, and ensuring consistent, reliable performance across prompts and model versions is essential. Extending these hybrid methods to other formal languages and enabling repair tools to learn from experience are important directions for future research. Our study highlights a promising future for specification repair that leverages both traditional and AI-driven approaches. By embracing this hybrid paradigm, we can work towards more robust, efficient, and accessible formal verification techniques that contribute to reliable and secure software systems.

VII. Conclusion and Future Work

This study presents a comprehensive empirical evaluation of Alloy specification repair techniques, exploring the synergy between traditional tools and emerging LLM-based approaches in the context of enhancing software dependability. Our findings offer significant insights into improving the reliability and correctness of formal specifications, a critical aspect of developing dependable software systems.

Key conclusions from our research include: (1) The complementary strengths of traditional and LLM-based repair techniques, with each excelling in different types of specification errors; (2) The superior performance of hybrid approaches that combine traditional tools with multi-round LLM techniques, achieving repair rates of up to 85.5% across benchmarks, thus significantly enhancing specification reliability; and (3) The potential for developing more robust and adaptive repair strategies that can significantly improve the dependability of Alloy specifications across various domains and complexity levels. While our evaluation is centered on Alloy, the observed trends have broader implications for formal specification repair in general. The structured nature of traditional techniques allows for precise constraint modification, whereas LLM-based methods introduce adaptability in handling underspecified or ambiguous constraints. These characteristics suggest that hybrid approaches could be effective across other declarative languages with similar specification repair challenges. Future work should explore their applicability to other specification languages to validate their generalizability.

Our work contributes to the field of dependable systems by providing empirical evidence for the effectiveness of integrated repair approaches in enhancing specification quality. We demonstrate that combining rule-based, systematic techniques with the flexible, context-aware capabilities of LLMs can significantly improve the dependability of formal specifications, a crucial foundation for reliable software systems.

Looking ahead, this research opens up several promising avenues for future work in dependable systems: (1) Developing more sophisticated hybrid repair systems that dynamically adapt to different specification types and error patterns, enhancing the overall reliability of formal modeling processes; (2) Investigating techniques to improve the interpretability and consistency of LLM-generated repairs, crucial for their adoption in safety-critical and high-assurance systems; (3) Exploring the application of these hybrid approaches to other formal specification languages used in dependable systems development; and (4) Addressing scalability challenges to ensure the effectiveness of these techniques for large-scale, complex specifications typical in critical infrastructure and mission-critical software.

In conclusion, our study underscores the potential for a paradigm shift in specification repair, moving towards more integrated, AI-assisted approaches that can significantly enhance the dependability of software systems. By continuing to explore and refine these hybrid methodologies, we can work towards more reliable, efficient, and robust formal verification techniques. This advancement promises to contribute substantially to the development of highly dependable software systems, crucial in an era where software reliability is paramount across various critical domains.

Data Availability

All data associated with this study are available at the following URL, which is a persistent repository:
https://doi.org/10.6084/m9.figshare.28636109

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments. This work was completed utilizing the Holland Computing Center of the University of Nebraska, which receives support from the Nebraska Research Initiative. This work was supported in part by awards CCF-1755890, CCF-1618132, CCF-2139845, and CCF-2124116 from the National Science Foundation.

REFERENCES

[1] D. Akhawe, A. Barth, P. E. Lam, J. Mitchell, and D. Song, "Towards a formal foundation of web security," in *2010 23rd IEEE Computer Security Foundations Symposium*, 2010, pp. 290–304.

[2] H. Bagheri and K. Sullivan, "Model-driven synthesis of formally precise, stylized software architectures," *Form. Asp. Comput.*, vol. 28, no. 3, pp. 441–467, May 2016.

[3] J. Hao, E. Kang, J. Sun, and D. Jackson, "Designing minimal effective normative systems with the help of lightweight formal methods," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2016, 2016, pp. 50–60.

[4] D. Jackson, *Software Abstractions - Logic, Language, and Analysis*. MIT Press, 2006.

[5] N. Mirzaei, J. Garcia, H. Bagheri, A. Sadeghi, and S. Malek, "Reducing combinatorics in gui testing of android applications," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 559–570. [Online]. Available: https://doi.org/10.1145/2884781.2884853

[6] H. Bagheri, E. Kang, and N. Mansoor, "Synthesis of assurance cases for software certification," in *ICSE-NIER 2020: 42nd International Conference on Software Engineering, New Ideas and Emerging Results, Seoul, South Korea, 27 June - 19 July, 2020*. ACM, pp. 61–64.

[7] H. Bagheri, C. Tang, and K. Sullivan, "Trademaker: automated dynamic analysis of synthesized tradespaces," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 106–116. [Online]. Available: https://doi.org/10.1145/2568225.2568291

[8] B. Schmerl, J. Gennari, A. Sadeghi, H. Bagheri, S. Malek, J. Cámara, and D. Garlan, "Architecture modeling and analysis of security in android systems," in *Software Architecture*, B. Tekinerdogan, U. Zdun, and A. Babar, Eds. Cham: Springer International Publishing, 2016, pp. 274–290.

[9] H. Bagheri, C. Tang, and K. Sullivan, "Automated synthesis and dynamic analysis of tradeoff spaces for object-relational mapping," *IEEE Transactions on Software Engineering*, vol. 43, no. 2, pp. 145–163, 2017.

[10] H. Bagheri, Y. Song, and K. Sullivan, "Architectural style as an independent variable," in *Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 159–162. [Online]. Available: https://doi.org/10.1145/1858996.1859026

[11] M. Alhanahnah, C. Stevens, B. Chen, Q. Yan, and H. Bagheri, "Iotcom: Dissecting interaction threats in iot systems," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 1523–1539, 2023.

[12] H. Bagheri, A. Sadeghi, R. Jabbarvand, and S. Malek, "Practical, formal synthesis and automatic enforcement of security policies for android," in *Proceedings of DSN*, 2016, pp. 514–525.

[13] H. Bagheri, E. Kang, S. Malek, and D. Jackson, "Detection of design flaws in the android permission protocol through bounded verification," in *FM 2015: Formal Methods*, N. Bjørner and F. de Boer, Eds. Cham: Springer International Publishing, 2015, pp. 73–89.

[14] H. Bagheri, J. Wang, J. Aerts, N. Ghorbani, and S. Malek, "Flair: efficient analysis of android inter-component vulnerabilities in response to incremental changes," *Empir. Softw. Eng.*, vol. 26, no. 3, p. 54, 2021.

[15] M. Alhanahnah, C. Stevens, and H. Bagheri, "Scalable analysis of interaction threats in iot systems," in *ISSTA'20: 29th ACM SIGSOFT*, pp. 272–285.

[16] H. Bagheri, J. Wang, J. Aerts, and S. Malek, "Efficient, evolutionary security analysis of interacting android apps," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2018, pp. 357–368.

[17] C. Stevens, M. Alhanahnah, Q. Yan, and H. Bagheri, "Comparing formal models of iot app coordination analysis," in *Proceedings of the 3rd ACM SIGSOFT International Workshop on Software Security from Design to Deployment*, ser. SEAD 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 3–10. [Online]. Available: https://doi.org/10.1145/3416507.3423188

[18] H. Bagheri and K. Sullivan, "Pol: specification-driven synthesis of architectural code frameworks for platform-based applications," *SIGPLAN Not.*, vol. 48, no. 3, p. 93–102, Sep. 2012. [Online]. Available: https://doi.org/10.1145/2480361.2371416

[19] S. Khurshid and D. Marinov, "Testera: Specification-based testing of java programs using sat," *Automated Software Engineering*, vol. 11, 2004.

[20] N. Mirzaei, J. Garcia, H. Bagheri, A. Sadeghi, and S. Malek, "Reducing combinatorics in gui testing of android applications," in *Proceedings ICSE*, 2016, pp. 559–570.

[21] N. Mansoor, J. A. Saddler, B. Silva, H. Bagheri, M. B. Cohen, and S. Farritor, "Modeling and testing a family of surgical robots: An experience report," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2018, 2018, pp. 785–790.

[22] N. Mansoor, H. Bagheri, E. Kang, and B. Sharif, "An empirical study assessing software modeling in alloy," in *2023 IEEE/ACM 11th International Conference on Formal Methods in Software Engineering (FormaliSE)*, 2023, pp. 44–54.

[23] A. Jovanovic and A. Sullivan, "Right or wrong – understanding how novice users write software models," 2024. [Online]. Available: https://arxiv.org/abs/2402.06624

[24] G. Zheng, T. Nguyen, S. Gutiérrez Brida, G. Regis, M. F. Frias, N. Aguirre, and H. Bagheri, "Flack: Counterexample-guided fault localization for alloy models," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 637–648.

[25] K. Wang, A. Sullivan, and S. Khurshid, "Automated model repair for alloy," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 577–588. [Online]. Available: https://doi.org/10.1145/3238147.3238162

[26] S. Gutiérrez Brida, G. Regis, G. Zheng, H. Bagheri, T. Nguyen, N. Aguirre, and M. Frias, "Icebar: Feedback-driven iterative repair of alloy specifications," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '22. New York, NY, USA: Association for Computing Machinery, 2023.

[27] S. G. Brida, G. Regis, G. Zheng, H. Bagheri, T. Nguyen, N. Aguirre, and M. Frias, "Bounded exhaustive search of alloy specification repairs," in *Proceedings of the 43rd International Conference on Software Engineering*, ser. ICSE '21. IEEE Press, 2021, p. 1135–1147.

[28] G. Zheng, T. Nguyen, S. G. Brida, G. Regis, N. Aguirre, M. F. Frias, and H. Bagheri, "Atr: Template-based repair for alloy specifications," in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 666–677.

[29] G. Zheng, T. Nguyen, S. Gutiérrez Brida, G. Regis, M. Frias, N. Aguirre, and H. Bagheri, "Flack: Localizing faults in alloy models," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 1218–1222.

[30] S. Gutiérrez Brida, G. Regis, G. Zheng, H. Bagheri, T. Nguyen, N. Aguirre, and M. Frias, "Artifact of bounded exhaustive search of alloy specification repairs," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2021, pp. 209–210.

[31] G. Zheng, H. Bagheri, and T. Nguyen, "Debugging declarative models in alloy," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2020, pp. 844–848.

[32] S. Gutiérrez Brida, G. Regis, G. Zheng, H. Bagheri, T. Nguyen, N. Aguirre, and M. Frias, "Beafix: An automated repair tool for faulty alloy models," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 1213–1217.

[33] M. R. Hasan, J. Li, I. Ahmed, and H. Bagheri, "Automated repair of declarative software specifications in the era of large language models," 2023.

[34] M. Alhanahnah, M. R. Hasan, and H. Bagheri, "An empirical evaluation of pre-trained large language models for repairing declarative formal specifications," *arXiv preprint arXiv:2404.11050*, 2024.

[35] K. Wang, A. Sullivan, and S. Khurshid, "Arepair: A repair framework for alloy," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2019, pp. 103–106.

[36] N. Macedo, A. Cunha, J. Pereira, R. Carvalho, R. Silva, A. C. Paiva, M. Sozinho Ramalho, and D. Silva, "Experiences on teaching alloy with an automated assessment platform," *Sci. Comput. Program.*, vol. 211, no. C, nov 2021. [Online]. Available: https://doi.org/10.1016/j.scico.2021.102690

[37] *D. Jackson, Software Abstractions*. MIT Press, 2012.

[38] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[39] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[40] E. Eid, "Profiling Alloy models," Master's thesis, University of Waterloo, 2021.

[41] T. Gärtner, P. A. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, ser. Lecture Notes in Computer Science, B. Schölkopf and M. K. Warmuth, Eds., vol. 2777. Springer, 2003, pp. 129–143. [Online]. Available: https://doi.org/10.1007/978-3-540-45167-9_11

[42] R. Torres, T. Ludwig, J. M. Kunkel, and M. F. Dolz, "Comparison of clang abstract syntax trees using string kernels," in *2018 International Conference on High Performance Computing & Simulation, HPCS 2018, Orleans, France, July 16-20, 2018*. IEEE, 2018, pp. 106–113. [Online]. Available: https://doi.org/10.1109/HPCS.2018.00032

[43] S. Masuda, T. Matsuodani, and K. Tsuda, "Syntax-tree similarity for test-case derivability in software requirements," in *14th IEEE International Conference on Software Testing, Verification and Validation Workshops, ICST Workshops 2021, Porto de Galinhas, Brazil, April 12-16, 2021*. IEEE, 2021, pp. 162–172. [Online]. Available: https://doi.org/10.1109/ICSTW52544.2021.00037

[44] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," *Noise reduction in speech processing*, pp. 1–4, 2009.

[45] D. Coughlin, "Correlating automated and human assessments of machine translation quality," in *Proceedings of Machine Translation Summit IX: Papers*, 2003.

[46] X. Hu, Q. Chen, H. Wang, X. Xia, D. Lo, and T. Zimmermann, "Correlating automated and human evaluation of code documentation generation quality," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 4, pp. 1–28, 2022.