

KIMI-DEV: AGENTLESS TRAINING AS SKILL PRIOR FOR SWE-AGENTS

Anonymous authors

Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) are increasingly applied to software engineering (SWE), with SWE-bench as a key benchmark. Solutions are split into SWE-Agent frameworks with multi-turn interactions and workflow-based Agentless methods with single-turn verifiable steps. We argue these paradigms are not mutually exclusive: reasoning-intensive Agentless training induces skill priors, including localization, code edit, and self-reflection that enable efficient and effective SWE-Agent adaptation. In this work, we first curate the Agentless training recipe and present Kimi-Dev, an open-source SWE LLM achieving 60.4% on SWE-bench Verified, the best among workflow approaches. With additional SFT adaptation on 5k publicly-available trajectories, Kimi-Dev powers SWE-Agents to 48.6% pass@1, on par with that of Claude 3.5 Sonnet (241022 version). These results show that structured skill priors from Agentless training can bridge workflow and agentic frameworks for transferable coding agents.

1 INTRODUCTION

Recent days have witnessed the rapid development of Large Language Models (LLMs) automating Software-Engineering (SWE) tasks (Jimenez et al., 2023; Yang et al., 2024a; Xia et al., 2024; Anthropic, 2024; Pan et al., 2024; Wang et al., 2025a; Wei et al., 2025; Yang et al., 2025a; Kimi et al., 2025; OpenAI, 2025c). Among the benchmarks that track the progress of LLM coding agents in SWE scenarios, SWE-bench (Jimenez et al., 2023) stands out as one of the most representative ones: Given an issue that reports a bug in a real-world GitHub repository, a model is required to produce a patch that fixes the bug, the correctness of which is further judged by whether the corresponding unit tests are passed after its application. The difficulty of the task (as of the date the benchmark was proposed), the existence of the outcome reward with the provided auto-eval harness, as well as the real-world economic value it reflects, have made the SWE-bench a focal point of the field.

Two lines of solutions have emerged for the SWE-bench task. Agent-based solutions like SWE-Agent (Yang et al., 2024a) and OpenHands (Wang et al., 2025a) take an interactionist approach: Instructed with the necessary task description, a predefined set of available tools, as well as the specific problem statement, the agent is required to interact with an executable environment for *multiple turns*, make change to the source codes, and determine when to stop autonomously. In contrast, workflow-based solutions like Agentless (Xia et al., 2024) pre-define the solving progress as a pipeline, which consists of steps like localization, bug repair, and test composition. Such task decomposition transforms the agentic task into generating correct responses for a chain of *single-turn* problems with verifiable rewards (Guo et al., 2025; Wei et al., 2025; He et al., 2025).

The two paradigms have been widely viewed as mutually exclusive. On the one hand, SWE-Agents are born with higher potential and better adaptability, thanks to the higher degree of freedom of the multi-turn interaction without the fixed routines. However, it has also proved more difficult to train with such frameworks due to their end-to-end nature (Luo et al., 2025; Cao et al., 2025). On the other hand, Agentless methods offer better modularity and the ease to train with Reinforcement Learning with Verifiable Rewards (RLVR) techniques, but more limited exploration space and flexibility, and difficulty in behavior monitoring as the erroneous patterns appear only in the single-turn long reasoning contents (Pan et al., 2024). However, we challenge the dichotomy from the perspective of training recipe: We argue that Agentless training should not be viewed as the ultimate deliverable, but rather as a way to induce skill priors – atomic capabilities such as the localization of buggy im-

plementations and the update of erroneous code snippets, as well as self-reflection and verification, all of which help scaffold the efficient adaptation of more capable and generalizable SWE-agents.

Guided by this perspective, we introduce Kimi-Dev, an open-source code LLM for SWE tasks. Specifically, we first develop an Agentless training recipe, which includes mid-training, cold-start, reinforcement learning, and test-time self-play. This results in 60.4% accuracy on SWE-bench Verified, the SoTA performance among the workflow-based solutions. Building on this, we show that Agentless training induces skill priors: a minimal SFT cold-start from Kimi-Dev with 5k publicly-available trajectories enables efficient SWE-agent adaptation and reaches 48.6% pass@1 score, similar to that of Claude 3.5 Sonnet (the 20241022 version, Anthropic (2024)). We demonstrate that these induced skills transfer from the non-agentic workflows to the agentic frameworks, and the self-reflection in long Chain-of-Thoughts baked through Agentless training further enable the agentic model to leverage more turns and succeed with a longer horizon. Finally, we also show that the skills from Agentless training generalize beyond SWE-bench Verified to broader benchmarks like SWE-bench-live (Zhang et al., 2025) and SWE-bench Multilingual (Yang et al., 2025c). Together, these results reframe the relationship between Agentless and agentic frameworks: not mutually exclusive, but as complementary stages in building transferable coding LLMs. This shift offers a principled view that training with structural skill priors could scaffold autonomous agentic interaction.

The remainder of this paper is organized as follows. Section 2 reviews the background of the framework dichotomy and outlines the challenges of training SWE-Agents. Section 3 presents our Agentless training recipe and the experimental results. Section 4 demonstrates how these Agentless-induced skill priors enable efficient SWE-Agent adaptation, and evaluates the skill transfer and generalization beyond SWE-bench Verified.

2 BACKGROUND

In this section, we first review the two dominant frameworks for SWE tasks and their dichotomy in Section 2.1. We then summarize the progress and challenges of training SWE-Agents in Section 2.2. The background introduction sets the stage for reinterpreting Agentless training as skill priors for SWE-Agents, a central theme developed throughout the later sections.

2.1 FRAMEWORK DICHOTOMY

Two paradigms currently dominate the solutions for automating software engineering tasks. Agentless approaches decompose SWE tasks into modular workflows (Xia et al., 2024; Wei et al., 2025; Ma et al., 2025a;b; Xie et al., 2025). Typical workflows consist of bug localization, bug repair, and test generation. This design provides modularity and stability: each step could be optimized separately as a single-turn problem with verifiable rewards (Wei et al., 2025; He et al., 2025). However, such rigidity comes at the cost of flexibility. When encountering scenarios requiring multiple rounds of incremental updates, the Agentless approaches struggle to adapt.

By contrast, SWE-agents adopt an end-to-end, multi-turn reasoning paradigm (Yang et al., 2024a; Wang et al., 2025a). Rather than following a fixed workflow, they iteratively plan, act, and reflect, resembling how human developers debug complex issues. This design enables greater adaptability, but introduces significant difficulties: trajectories often extend over tens or even hundreds of steps, context windows of the LLMs must span over the entire interaction history, and the model must handle exploration, reasoning, and tool use simultaneously.

The dichotomy between fixed workflows (*e.g.*, Agentless) and agentic frameworks (*e.g.*, SWE-Agent) has shaped much of the community’s perspective. The two paradigms are often regarded as mutually exclusive: one trades off flexibility and performance ceiling for modularity and stability, whereas the other makes the reverse compromise. Our work challenges this dichotomy, as we demonstrate that Agentless training induces skill priors that make further SWE-agent training both more stable and more efficient.

2.2 TRAINING SWE-AGENTS

Training SWE-agents relies on acquiring high-quality trajectories through interactions with executable environments. Constructing such large-scale environments and collecting reliable trajec-

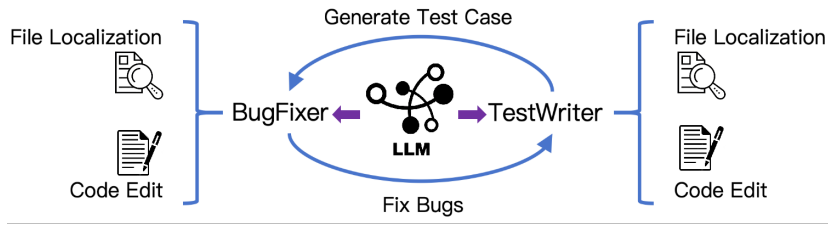


Figure 1: Agentless framework for Kimi-Dev: the duo of BugFixer and TestWriter.

ries, however, requires substantial human labor as well as costly calls to frontier models, making data collection slow and resource-demanding (Pan et al., 2024; Badertdinov et al., 2024b). Recent studies also attempt to scale environment construction by synthesizing bugs for the reverse construction of executable runtime (Jain et al., 2025; Yang et al., 2025c). However, credit assignment across long horizons still remains challenging, as outcome rewards are sparse and often only available when a final patch passes its tests. Reinforcement learning techniques have been proposed, but frequently suffer from instability or collapse when trajectories exceed dozens of steps (Luo et al., 2025; Cao et al., 2025). SWE-agent training is also highly sensitive to initialization: starting from a generic pre-trained model often leads to brittle behaviors, such as failing to use tools effectively or getting stuck in infinite loops of specific action patterns (Pan et al., 2024; Yang et al., 2025c).

These limitations motivate our central hypothesis: instead of training SWE-agents entirely from scratch, one can first induce skill priors through agentless training, enhancing the atomic capabilities like localization, repair, test composition, and self-reflection. These priors lay a foundation that makes subsequent agentic training both more efficient and more generalizable.

3 AGENTLESS TRAINING RECIPE

Instead of training SWE-agents from scratch, we leverage Agentless training to induce skill priors. Skill priors enhanced by Agentless training include but are not limited to bug localization, patch generation, self-reflection and verification, which lay the foundation for end-to-end agentic interaction. In this section, we elaborate our Agentless training recipe: the duo framework design of BugFixer and TestWriter, mid-training and cold-start, reinforcement learning, and test-time self-play. Sections 3.1–3.4 detail these ingredients, and Section 3.5 presents the experimental results for each of them. This training recipe results in Kimi-Dev, an open-source 72B model that achieves 60.4% on SWE-bench Verified, the SoTA performance among the workflow-based solutions.

3.1 FRAMEWORK: THE DUO OF BUGFIXER AND TESTWRITER

In GitHub issue resolution, we conceptualize the process as the collaboration between two important roles: the BugFixer, who produces patches that correctly address software bugs, and the TestWriter, who creates reproducible unit tests that capture the reported bug. A resolution is considered successful when the BugFixer’s patch passes the tests provided for the issue, while a high-quality test from the TestWriter should fail on the pre-fix version of the code and pass once the fix is applied.

Each role relies on two core skills: (i) file localization, the ability to identify the specific files relevant to the bug or test, and (ii) code edit, the ability to implement the necessary modifications. For BugFixer, effective code edits repair the defective program logic, whereas for TestWriter, they update precise unit test functions that reproduce the issue into the test files. As illustrated in Figure 1, these two skills constitute the fundamental abilities underlying GitHub issue resolution. Thus, we enhance these skills through the following training recipes, including mid-training, cold-start, and RL.

3.2 MID-TRAINING & COLD START

To enhance the model’s prior as both a BugFixer and a TestWriter, we perform mid-training with $\sim 150\text{B}$ tokens in high-quality and real-world data. With the Qwen 2.5-72B-Base (Qwen et al., 2024) model as a starting point, we collect millions of GitHub issues and PR commits to form its mid-training dataset, which consists of (i) $\sim 50\text{B}$ tokens in the form of Agentless derived from the

natural diff patch, (ii) $\sim 20\text{B}$ tokens of curated PR commit packs, and (iii) $\sim 20\text{B}$ tokens of synthetic data with reasoning and agentic interaction patterns (upsampled by a factor of 4 during training). The data recipe is carefully constructed to enable the model to learn how human developers reason with GitHub issues, implement code fixes, and develop unit tests. We also performed strict data decontamination to exclude any repository from the SWE-bench Verified test set. Mid-training sufficiently enhances the knowledge in the model about practical bug fixes and unit tests, making it a better starting point for later stages. The details of the recipe are covered in Appendix A.

To activate the model’s long Chain-of-Thought (CoT) capability, we also construct a cold-start dataset with reasoning trajectories based on the SWE-Gym (Pan et al., 2024) and SWE-bench-extra (Badertdinov et al., 2024a) datasets, generated by the DeepSeek R1 model (Luo et al. (2025), the 20250120 version). In this setup, R1 acts the roles of Bugfixer and Testwriter, producing outputs such as file localization and code edits. Through supervised finetuning as a cold start with this dataset, we enable the model to acquire essential reasoning skills, including problem analysis, method sketching, self-refinement, and exploration of alternative solutions.

3.3 REINFORCEMENT LEARNING

After mid-training and cold-start, the model demonstrates strong performance in localization. Therefore, reinforcement learning (RL) focuses solely on the code edit stage. We construct a training set specifically for this stage, where each prompt is equipped with an executable environment. We further employ multiple localization rollouts from the initial model to generate varied file location predictions, which diversifies the prompts used in code-edit RL.

For the RL algorithm, we adopt the policy optimization method proposed by Kimi k1.5 (Team et al., 2025), which has shown promising results on reasoning tasks in both math and coding. Unlike GRPO (Shao et al., 2024) that eliminates the critic by using average rewards for advantage estimation, Kimi k1.5 (Team et al., 2025) adopts a simpler policy gradient approach motivated by path consistency (Nachum et al., 2017). The specific RL objective is provided in Appendix C.1.

When adapting the algorithm in our SWE-bench setting, we highlight the following 3 key desiderata:

1. **Outcome-based reward only:** We rely solely on the final execution outcome from the environment as the raw reward (0 or 1), without incorporating any format- or process-based signals. For BugFixer, a positive reward is given if the generated patch passes all ground-truth unittests. For TestWriter, a positive reward is assigned when (i) **the predicted test raises a failure in the repo without the ground-truth bugfix patch applied, AND (ii) the failure is resolved once the ground-truth bugfix patch is applied.**
2. **Adaptive prompt selection:** Prompts with $\text{pass}@16 = 0$ are initially discarded as they do not contribute to the batch loss. **This results in an initial prompt set of 1,200 problems and** enlarges the effective batch size. A curriculum learning scheme is then applied: once the success rate on the current set exceeds a threshold, **500 new** (previously excluded) prompts (with initial $\text{pass}@16 = 0$ but improved under RL) are reintroduced **every 100 RL steps** to gradually raise task difficulty.
3. **Positive example reinforcement:** As performance improvements begin to plateau in later stages of training, we incorporate the positive samples from the recent RL iterations into the training batch of the current iteration. This approach reinforces the model’s reliance on successful patterns, thereby accelerating convergence in the final phase.

Robust sandbox infrastructure. We construct the docker environment with Kubernetes (Burns et al., 2016), which provides a secure and scalable sandbox infrastructure and efficient training and rollouts. The infra supports over 10,000 concurrent instances with robust performance, making it ideal for competitive programming and software engineering tasks (see Appendix D).

3.4 TEST-TIME SELF-PLAY

After RL, the model masters the roles of both a BugFixer and a TestWriter. During test time, it adopts a self-play mechanism to coordinate its bug-fixing and test-writing abilities. Following Agentless (Xia et al., 2024), we leverage the model to generate 40 candidate patches and 40 tests for each instance. Each patch generation involves independent runs of the localization and code edit

Table 1: Performance comparison for models on SWE-bench Verified under Agentless-like frameworks. All the performances are obtained under the standard 40 patch, 40 test setting (Xia et al., 2024), except that Llama3-SWE-RL uses 500 patches and 30 tests.

Model	#Params	Resolve Rate (%)
Llama3-SWE-RL (Wei et al., 2025)	70B	41.0
Seed1.5-Thinking (Seed et al., 2025)	200B	47.0
OpenAI-o1 (OpenAI, 2024)	-	48.9
DeepSeek-R1-0120 (Guo et al., 2025)	671B	49.2
OpenAI-o3-mini-high (OpenAI, 2025a)	-	49.3
Claude 3.5 Sonnet (241022) (Anthropic, 2024)	-	50.8
MiniMax-M1 (Chen et al., 2025a)	456B	56.0
DeepSeek-R1-0528 (Guo et al., 2025)	671B	57.6
SWE-SWISS (He et al., 2025)	32B	58.2
Kimi-Dev (Ours)	72B	60.4

from BugFixer, where the first run uses greedy decoding (temperature 0), and the remaining 39 use temperature 1 to ensure diversity. Similarly, 40 tests are generated independently from TestWriter. For the test patch candidates, to guarantee their validity, we first filter out those failing to raise a failure in the original repo without applying any BugFixer patch.

Denote the rest TestWriter patches as set \mathcal{T} , and the BugFixer patches as set \mathcal{B} . For each $b_i \in \mathcal{B}$ and $t_j \in \mathcal{T}$, we execute the test suite over the test file modified by t_j for twice: first without b_i , and then with b_i applied. From the execution log for the first run, we get the count of the failed and the passed tests from t_j , denoted as $F(j)$ and $P(j)$. Comparing the execution logs for the two test suite runs, we get the count of the fail-to-pass and the pass-to-pass tests, denoted as $FP(i, j)$ and $PP(i, j)$, respectively. We then calculate the score for each b_i with

$$S_i = \frac{\sum_j FP(i, j)}{\sum_j F(j)} + \frac{\sum_j PP(i, j)}{\sum_j P(j)}, \quad (1)$$

where the first part reflects the performance of b_i under reproduction tests, and the second part could be viewed as the characterization of b_i under regression tests (Xia et al., 2024). We select the BugFixer patch b_i with the highest S_i score as the ultimate answer.

3.5 EXPERIMENTS

3.5.1 MAIN RESULTS

We evaluate our approach on SWE-bench Verified (Jimenez et al., 2023), which includes 500 GitHub issues guaranteed to be solved by human programmers. Table 1 shows the overall results. Unlike SWE-RL (Wei et al., 2025), which relies on text-based similarity to the ground-truth patch, we use execution results as the reward signal, reducing shortcuts and improving solution quality. Unlike most Agentless approaches (Xia et al., 2024; Guo et al., 2025; He et al., 2025), which place a single-file reproduction test at the root directory of the repository, we leverage the two-stage TestWriter to better capture repository context and align with human standard workflows (OpenAI, 2025). Kimi-Dev model achieves the state-of-the-art performance among open-source models, resolving 60.4% of the issues in SWE-bench Verified.

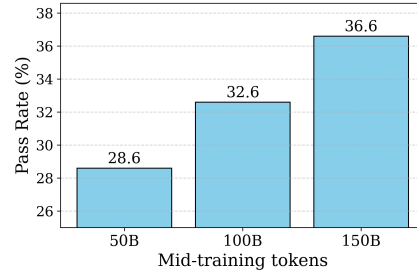


Figure 2: The performance on SWE-bench Verified after mid-training with different training token budgets.

3.5.2 MID-TRAINING

In this section, we evaluate the relationship between the amount of data used during mid-training and model performance. Specifically, we finetuned Qwen 2.5-72B-Base with the subset of mid-training data of 50B, 100B, and approximately 150B tokens, and then lightly activated them using

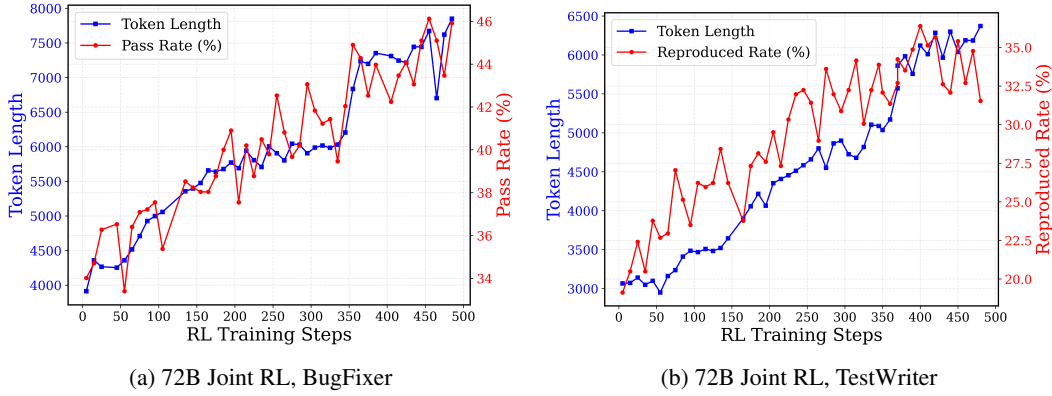


Figure 3: Joint code-edit RL experiments on the model after mid-training and cold-start. The pass rate for BugFixer and the reproduced rate for TestWriter are reported as pass@1 with temperature=1.0. The performance improves consistently as the output becomes increasingly longer.

the same 2,000 Bugfixer input-output pairs as cold start. We report BugFixer pass@1 without self-play for simplicity of evaluation. Figure 2 shows that increasing the number of tokens in mid-training consistently improves model performance, highlighting the effectiveness of this stage.

3.5.3 REINFORCEMENT LEARNING

Experimental setup We set the training step per RL iteration as 5 and sample 10 rollouts for each of the 1,024 problems from the union of SWE-gym (Pan et al., 2024) and SWE-bench-extra (Bardetdinov et al., 2024b). We dynamically adjust the prompt set every 20 iterations to gradually increase task difficulty. We fix the maximum training context length as 64k tokens, since the prompt input contains the contents of the entire files localized by the initial model in advance.

Results Figure 3 shows the performance and response length curves on the test set during RL training. The pass rate and the reproduced rate are calculated from pass@1 and temperature=1. Specifically, we observe that both model performance and response length steadily increase, reflecting the expected benefits of RL scaling. Similar RL scaling curves are also observed in our ablation experiments run on Qwen2.5-14B-Instruct models, proving the effectiveness of the RL training recipe across models of different sizes. The experimental details, as well as the ablation studies on positive example reinforcement in Section 3.3, are listed in Appendix C.3). The lengthy outputs consist of in-depth problem analysis and self-reflection patterns, similar to those in the math and code reasoning tasks (Team et al., 2025; Guo et al., 2025). We have also observed that for TestWriter, occasional false-positive examples take place during RL training due to the lack of reproduction coverage. We leave the case studies in Appendix F and further improvement for future work.

3.5.4 TEST-TIME SELF-PLAY

Following Section 3.4, we evaluate how the final performance on the SWE-bench Verified scales with the number of patches and tests generated. The temperature is fixed at 0 for the initial rollout and set to 1.0 for the subsequent 39 rollouts. As shown on the left of Figure 4, performance improves from 48.0% to 60.4% as the number of patch test pairs increases from 1×1 to 40×40 , and consistently outperforms the results obtained from the majority vote of BugFixer patches. Specifically, the self-play result obtained from 3 patches and 3 tests for each instance has already surpassed the performance with

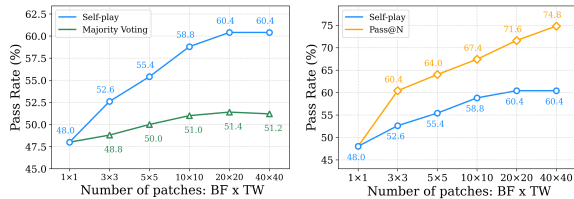


Figure 4: Test-time self-play on SWE-bench Verified. Performance improves with more generated patches and tests. Left: Execution-based self-play consistently surpasses BugFixer majority voting. Right: Self-play performances remain below pass@N where the ground-truth test patch is used, suggesting the room exists for TestWriter to improve.

Table 2: Single-attempt performance of different models on SWE-bench Verified under end-to-end agentic frameworks, categorized by proprietary or open-weight models, and size over or under 100B. “Internal” denotes results achieved with their in-house agentic frameworks.

Model	System	#Params	Pass Rate (%)
<i>Proprietary</i>			
Gemini 2.5 Pro (Comanici et al., 2025)	Internal	-	60.3
OpenAI-o3 (OpenAI, 2025)	Internal	-	69.1
GPT-5 (OpenAI, 2025c)	Internal	-	74.9
Claude 3.5 Sonnet (241022) (Anthropic, 2024)	SWE-Agent	-	49.0
Claude 3.7 Sonnet (Anthropic, 2025a)	SWE-Agent	-	62.3
Claude 4.0 Sonnet (Anthropic, 2025b)	SWE-Agent	-	72.7
<i>Open Weight, $\geq 100B$</i>			
gpt-oss-120b (High) (OpenAI, 2025b)	Internal	120B	62.4
DeepSeek-v3.1 (Guo et al., 2025)	Internal	671B	66.0
Kimi-K2-0905 (Kimi et al., 2025)	SWE-Agent	1T	69.2
Qwen3-Coder (Yang et al., 2025a)	OpenHands	480B	69.6
<i>Open Weight, $< 100B$</i>			
Openhands-LM (Wang et al., 2025b)	OpenHands	32B	37.2
Skywork-SWE (Zeng et al., 2025)	OpenHands	32B	38.0
SWE-agent-LM (Yang et al., 2025b)	SWE-Agent	32B	40.2
DeepSWE (Luo et al., 2025)	OpenHands	32B	42.2
Devstral-Small-2507 (AI & AI, 2025)	OpenHands	24B	53.6
gpt-oss-20b (High) (OpenAI, 2025b)	Internal	20B	60.7
Kimi-Dev (SFTed)	SWE-Agent	72B	48.6

majority voting from 40 BugFixer patches. This demonstrates the effectiveness of additional information from test-time execution. The room for improvement of TestWriter, though, still exists for more powerful self-play: Shown on Figure 4, self-play performances remain below pass@N, where ground-truth test cases serve as the criterion for issue resolution. This finding aligns with Anthropic (2024), which introduced a final edge-case checking phase to generate a more diverse set of test cases, thereby strengthening the role of the “TestWriter” in their SWE-Agent framework. We also report preliminary observations of a potential parallel scaling phenomenon, which requires no additional training and may enable scalable performance improvements (see Appendix G).

4 INITIALIZING SWE-AGENTS FROM AGENTLESS TRAINING

End-to-end multi-turn frameworks, such as SWE-Agent (Yang et al., 2024a; Anthropic, 2024) and OpenHands (Wang et al., 2025a), enable agents to leverage tools and interact with environments. Specifically, the system prompt employed in the SWE-Agent framework (Anthropic, 2024) outlines a five-stage workflow: (i) repo exploration, (ii) error reproduction via a test script, (iii) code edit for bug repair, (iv) test re-execution for validation, and (v) edge-case generation and checks. Unlike Agentless, the SWE-Agent framework doesn’t enforce a strict stage-wise workflow; the agent can reflect, transition, and redo freely until it deems the task complete and submits.

The performance potential is therefore higher without a fixed routine; However, the training for SWE-Agent is more challenging because of the sparsity of the outcome reward for long-horizon credit assignment. Meanwhile, our Kimi-Dev model has undergone Agentless training, with its skills of localization and code edit for BugFixer and TestWriter strengthened elaborately. In this section, we investigate whether it can serve as an effective prior for multi-turn SWE-Agent scenarios.

4.1 PERFORMANCE AFTER SWE-AGENT FINE-TUNING

We use the publicly available SWE-Agent trajectories to finetune Kimi-Dev. The finetuning dataset we used is released by SWE-smith (Yang et al., 2025b), consisting of 5,016 SWE-Agent trajectories collected with Claude 3.7 Sonnet (Anthropic, 2025a) in the synthetic environments. We perform

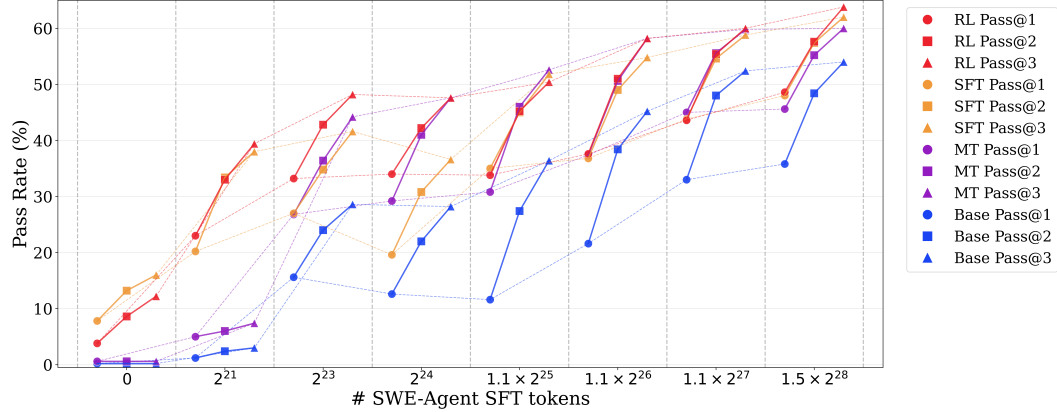


Figure 5: Comparing the quality of the raw Base, the Agentless mid-trained (MT), the Agentless mid-trained with reasoning-intensive cold-start (SFT), and the Kimi-Dev model after RL as the prior for SWE-Agent adaptation. The tokens of the SWE-Agent SFT trajectories are swept over different scales, and the SWE-Agent performances are reported up to pass@3 on SWE-bench Verified.

supervised fine-tuning over Kimi-Dev, setting the maximum context length as 64K tokens during training, and allowing up to 128K tokens and 100 turns during inference.

As shown in Table 2, without collecting more trajectory data over realistic environments, or conducting additional multi-turn agentic RL, our finetuned model achieves a pass@1 score of 48.6% on SWE-bench Verified under the agentic framework setup, without additional test-time scaling. Using the same SFT data, our finetuned Kimi-Dev model outperforms the SWE-agent-LM (Yang et al., 2025c), with the performance comparable to that of Claude 3.5 Sonnet (49% by the 241022 version). The pass@10 of our SWE-Agent adapted model is 74.0% and surpasses the pass@30 of our model under Agentless (73.8%), proving the higher potential for the SWE-Agent framework.

4.2 SKILL TRANSFER AND GENERALIZATION

The results shown in Section 4.1 demonstrate that Kimi-Dev, a model with extensive Agentless training, could be adapted to end-to-end SWE-Agents with lightweight supervised finetuning. As the Agentless training recipe consists of mid-training, cold-start (SFT) and RL, we explore the contribution of each part in the recipe to the SWE-Agent capability after adaptation.

To figure this out, we perform SWE-Agent SFT on the original Qwen2.5-72B (Base), the mid-trained model (MT), the model then activated with Agentless-formatted long CoT data (SFT), and the (Kimi-Dev) model after finishing RL training (RL). As we are treating the four models as the *prior* for SWE-Agents¹, and a good prior always demonstrates the ability of fast adaptation with a few shots (Finn et al., 2017; Brown et al., 2020), we also sweep the amount of SWE-Agent SFT data to measure the *efficiency* of each prior in SWE-Agent adaptation.

Specifically, we randomly shuffle the 5,016 SWE-Agent trajectories and construct nested subsets of sizes 100, 200, 500, 1,000, and 2,000, where each smaller subset is contained within the larger ones. In addition, we prepend two extreme baselines: (i) zero-shot, where the prior model is directly evaluated under the SWE-Agent framework without finetuning, and (ii) one-step gradient descent, where the model is updated with a single gradient step using the 100-trajectory subset. This yields a range of SFT token budgets spanning $\{0, 2^{21}, 2^{23}, 2^{24}, 1.1 \times 2^{25}, 1.1 \times 2^{26}, 1.1 \times 2^{27}, 1.5 \times 2^{28}\}$. After these lightweight SFT experiments, we evaluate performance in terms of pass@ $\{1, 2, 3\}$ under the SWE-Agent framework, with evaluations for pass@1 conducted at temperature 0, and those for pass@2 and pass@3 at temperature 1.0.

Figure 5 presents the SWE-Agent performances of each prior (Base, MT, SFT, RL) after being fine-tuned with different amounts of agentic trajectories. We have the following observations:

¹We slightly abuse the term “prior” to refer to a model to be finetuned with SWE-Agent trajectories in the following analysis.

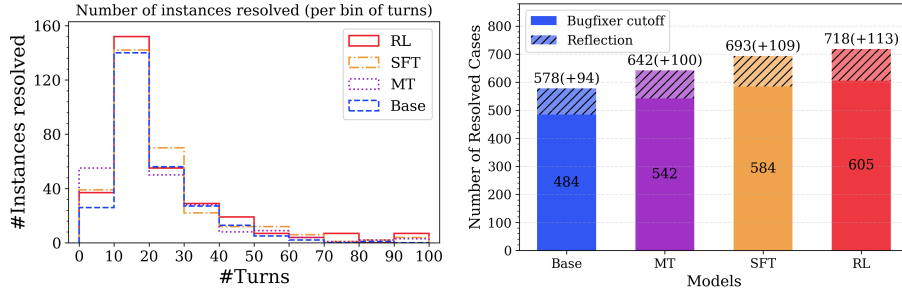


Figure 6: Left: Performance of the four priors under turn limits after SWE-Agent adaptation. Right: The characterization of the BugFixer and the reflection skills for each prior by counting the resolved cases of the 3 runs at Stage-3 cutoff moment, and comparing those with the final success cases.

1. The RL prior **outperforms** all the other models in nearly all the SWE-Agent SFT settings. This demonstrates that the Agentless training recipe indeed strengthens the prior in terms of SWE-Agent adaptation. For example, To achieve the top pass@1 performance of the Base prior, the RL prior needs only 2^{23} SWE-Agent SFT tokens, whereas the Base prior consumes 1.5×2^{28} tokens.
2. The MT prior is lagged behind the SFT and the RL ones in extremely data-scarce settings (zero-shot (0) and one-step gradient descent (2^{21})), but quickly becomes on par with them after 200 trajectories (2^{24}) are available for finetuning. This indicates that adaptation efficiency remains comparable after the prior is strengthened through Agentless mid-training.
3. The performance of the SFT prior is mostly similar to the RL one except for two cases: (i) The SFT prior outperforms the RL one under the zero-shot setting. This is reasonable, as the RL prior might overfit to the Agentless input-output format, while the SFT prior suffers less from this. (ii) The SFT prior exhibits a significant degradation with 200 SWE-Agent trajectories (2^{24}). A potential reason could be that the 200 trajectories collapse onto a single data mode, leading the SFT prior to overfit through memorization (Chu et al., 2025); the RL prior instead embeds stronger transferable skills and thus generalizes better.

From long CoT to extended multi-turn interactions. We hypothesize that reflective behaviors cultivated through long chain-of-thought reasoning may transfer to settings requiring extended multi-turn interactions. To examine this, we evaluate the four priors (Base, MT, SFT, and RL) by finetuning on the 5,016 trajectories and test on SWE-bench Verified, under varying turn limits with pass@3 as the metric (Figure 6, left). The distinct interaction-length profiles show supportive evidence: the RL prior, after finetuning, continues to make progress beyond 70 turns, while the SFT, mid-trained, and raw models show diminishing returns around 70, 60, and 50 turns, respectively.

We further evaluate the efficacy of the Agentless skill priors (**BugFixer** and **reflection**) in the SWE-Agent adapted model. For **BugFixer**, given that the SWE-Agent may autonomously reflect between the five stages, we examine the moment in each trajectory when the bug fix of the third stage is *initially* completed, and the test rerun of the fourth stage has not yet been entered. Heuristically, when the SWE-Agent just completes the third stage, it has not yet obtained the execution feedback from the fourth stage, and thus has not further reflected based on the execution information or refined the bug fix. We therefore calculate the success rate of direct submission at this cutoff moment, which reflects the capability of the BugFixer skill. Regarding **reflection**, we further compare the performance at the cutoff point with the performance after full completion for each problem. The increment in the number of successful problems is used to reflect the capability of the reflection skill.

We use kimi-k2-0711-preview (Kimi et al., 2025) to annotate the SWE-Agent trajectories, identifying the stage to which each turn belongs. Figure 6 (right) demonstrates that both skills are strengthened through each stage of the Agentless training recipe: For the BugFixer skill, the cutoff performance at Stage-3 within the SWE-Agent interaction trajectories of the four adapted models shows consistent improvement, ranging from 484 cases resolved by the Base prior to 605 cases by the RL prior, as measured by the number of successful resolutions within three passes. For the reflection skill, examining the performance gains from Stage-3 to the end of the trajectories reveals a similar trend, with improvements increasing from +94 under the Base prior to +113 under

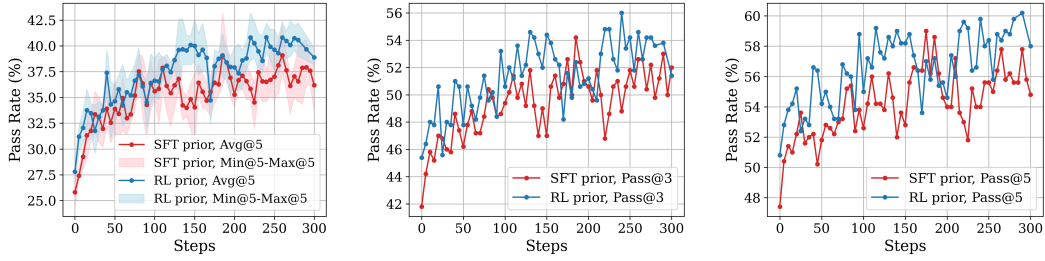


Figure 7: Comparison between the **SFT Prior** and the **RL Prior** when further applied with end-to-end SWE-Agent RL. Left: Pass@1 averaged from 5 runs. Middle: Pass@3. Right: Pass@5. The two priors are activated with the same 2^{21} SWE-Agent SFT tokens (the second column in Figure 5). After end-to-end RL, the RL prior slightly outperforms the SFT prior in all the Pass@1, Pass@3, and Pass@5 settings, which agrees with their SWE-Agent SFT performance comparison in Figure 5.

the RL prior. Taken together, the adapted model from the RL prior achieves the strongest overall performance across both skills. It should be noted that our analysis of the reflection skill remains coarse-grained, since the measured performance gains between the two checkpoints capture not only agentic reflection and redo behaviors, but also the intermediate test-writing process performed by the SWE-Agent. A more fine-grained evaluation that isolates the TestWriter skill prior is left for future work. The prompt for SWE-Agent stage annotation, extended qualitative studies, as well as additional discussions for skill transfer and generalization, are covered in Appendix E.

End-to-end SWE-Agent RL for prior comparison. To further validate the effectiveness of the priors baked by the Agentless training recipes, we employ end-to-end SWE-Agent RL (Luo et al., 2025) with the cold-started priors as the initial models. To maximally alleviate the effect from the patterns of proprietary models within the SWE-Smith trajectories, we leverage the setting with 2^{21} SWE-Agent SFT tokens, the second column in Figure 5, where a single step of gradient decent takes place on top of each prior. Under the minimal cold-start setup, end-to-end RL reveals the potential of each prior beyond taking the shortcut of imitation (Gudibande et al., 2024; Chu et al., 2025).

To run the end-to-end RL training for prior comparison, we use the SWE-Gym (Pan et al., 2024) and the SWE-bench-extra (Bardetdinov et al., 2024a) subsets as the training set. Similarly to the Agentless RL recipe, we first use each initial model to filter out the problems with Pass@8 = 0. For the model with the MT prior, 260 out of 6,202 problems remain; for the models with the SFT prior and the RL prior, a total of 2,062 from the 6,202 problems are kept. In all end-to-end RL runs, we use the outcome reward only, and the same policy gradient algorithm in Sec. 3.3 without KL or entropy regularization for optimization, with batch size as 256. The results are shown as follows:

For the model with MT prior, the pass@1 performance quickly deteriorates to less than 2% after 10 end-to-end RL steps. The potential reason for this could be the lack of available problems to be trained with, reflecting the inferiority of the prior. For the models with the SFT prior and the RL prior, the RL runs last for 300 steps, and we plot the performance comparison in Figure 7. According to Figure 7, the model with the RL prior demonstrates slightly higher scores of Pass@1, Pass@3, and Pass@5 over the model with the SFT prior. While the phenomenon agrees with the performance comparison under SWE-Agent SFT shown in Figure 5, we observe that the patterns in the interaction trajectories of the models incentivized by end-to-end SWE-Agent RL significantly differ from the patterns of the proprietary models (detailed in Appendix E). These results reveal that the Agentless training recipe curates strong priors for end-to-end learning under SWE-Agent frameworks with the minimal supervision of proprietary end-to-end trajectories. We leave the exploration of more advanced agentic RL techniques for further improvement as future work.

5 CONCLUSION AND FUTURE WORK

In this work, we reframed Agentless and agentic paradigms for automated software engineering as complementary rather than competing. By introducing Kimi-Dev, we demonstrated that structured Agentless training can induce transferable skill priors, including bug localization, code repair, and self-reflection. As a result, Kimi-Dev not only achieves SoTA results on SWE-bench Verified among the workflow-based approaches, but enables efficient SWE-Agent adaptation as well. These findings establish a novel path toward building more generalizable coding agents through staged training.

ETHICS AND REPRODUCIBILITY STATEMENTS

This work obeys the Code of Ethics required by the ICLR conference. The study does not involve human subjects or animal experimentation. The personally identifiable information from raw data is excluded for privacy consideration (see the mid-training data recipe detailed in Appendix A).

For all of the experiments, we have covered the detailed setups in the appendices: mid-training for Agentless in Appendix A, details of the used dockers in Appendix B, Agentless RL in Appendix C, agent infrastructure in Appendix D, and case studies in Appendix F.

REFERENCES

- Mistral AI and All Hands AI. Devstral-small-2507. <https://mistral.ai/news/devstral-2507>, July 2025.
- Anthropic. Raising the bar on swe-bench verified with claude 3.5 sonnet. Online; AI model, Oct 2024. URL <https://www.anthropic.com/engineering/swe-bench-sonnet>.
- Anthropic. Claude 3.7 sonnet: Hybrid reasoning model. <https://www.anthropic.com/news/claude-3-7-sonnet>, February 2025a.
- Anthropic. Claude sonnet 4. <https://www.anthropic.com/news/claude-4>, May 2025b.
- Ibragim Badertdinov, Maria Trofimova, Yuri Anapolskiy, Sergey Abramov, Karina Zainullina, Alexander Golubev, Sergey Polezhaev, Daria Litvintseva, Simon Karasik, Filipp Fisin, et al. Scaling data collection for training software engineering agents. *Nebius blog*, 2024a.
- Ibragim Badertdinov, Maria Trofimova, Yuri Anapolskiy, Sergey Abramov, Karina Zainullina, Alexander Golubev, Sergey Polezhaev, Daria Litvintseva, Simon Karasik, Filipp Fisin, Sergey Skvortsov, Maxim Nekrashevich, Anton Shevtsov, and Boris Yangel. Scaling data collection for training software engineering agents. *Nebius blog*, 2024b.
- Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. Swe-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents, 2025. URL <https://arxiv.org/abs/2505.20411>.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020. URL <https://arxiv.org/abs/2005.14165>.
- Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, Omega, and Kubernetes. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI ’16)*, pp. 285–301, 2016.
- Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, Shu Liu, Eric Tang, Jiayi Pan, Xingyao Wang, Akshay Malik, Graham Neubig, Kourosh Hakhmaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-v0: Train real-world long-horizon agents via reinforcement learning, 2025.
- Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025a.
- Mouxian Chen, Binyuan Hui, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Jianling Sun, Junyang Lin, and Zhongxin Liu. Parallel scaling law for language models. *arXiv preprint arXiv:2505.10475*, 2025b.

- Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *Forty-second International Conference on Machine Learning*, 2025. URL <https://openreview.net/forum?id=dYur3yabMj>.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Xinran Gu, Kaifeng Lyu, Jiazheng Li, and Jingzhao Zhang. Data mixing can induce phase transitions in knowledge acquisition. *arXiv preprint arXiv:2505.18091*, 2025.
- Arnav Gudibande, Eric Wallace, Charlie Victor Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The False Promise of Imitating Proprietary Language Models. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=Kz3yckpCN5>.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Zhenyu He, Qingping Yang, Wei Sheng, Xiaojian Zhong, Kechi Zhang, Chenxin An, Wenlei Shi, Tianle Cai, Di He, Jiaze Chen, Jingjing Xu, and Mingxuan Wang. Swe-swiss: A multi-task fine-tuning and rl recipe for high-performance issue resolution. Notion page, 2025.
- Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights swe agents. *arXiv preprint arXiv:2504.07164*, 2025.
- Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.
- Team Kimi, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.
- Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Ameen Patel, Qingyang Wu, Alpay Ariyak, Colin Cai, Shang Zhu Tarun Venkat, Ben Athiwaratkun, Manan Roongta, Ce Zhang, Li Erran Li, Raluca Ada Popa, Koushik Sen, and Ion Stoica. DeepSWE: Training a State-of-the-Art Coding Agent from Scratch by Scaling RL. Notion page, 2025. Notion Blog.
- Yingwei Ma, Rongyu Cao, Yongchang Cao, Yue Zhang, Jue Chen, Yibo Liu, Yuchen Liu, Binhua Li, Fei Huang, and Yongbin Li. Swe-gpt: A process-centric language model for automated software improvement. *Proc. ACM Softw. Eng.*, 2(ISSTA), June 2025a. doi: 10.1145/3728981. URL <https://doi.org/10.1145/3728981>.
- Yingwei Ma, Qingping Yang, Rongyu Cao, Binhua Li, Fei Huang, and Yongbin Li. Alibaba lingmaagent: Improving automated issue resolution via comprehensive repository exploration. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pp. 238–249, 2025b.

- Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/fac9f743b083008a894eee7baa16469-Paper.pdf.
- OpenAI. Openai o1 system card. Technical report, OpenAI, 2024. URL <https://arxiv.org/abs/2412.16720>. Includes o1 and o1-mini models, safety and evaluation work.
- OpenAI. Introducing codex. <https://openai.com/index/introducing-codex/>, May 2025.
- OpenAI. Openai o3 and openai o4-mini system card. Technical report, OpenAI, Apr 2025. URL <https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf>.
- OpenAI. Openai o3-mini system card. Technical report / system card, OpenAI, 2025a. URL <https://cdn.openai.com/o3-mini-system-card-feb10.pdf>. Description of o3-mini model, its safety evaluations and reasoning benchmarks.
- OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025b. URL <https://arxiv.org/abs/2508.10925>.
- OpenAI. Gpt-5 system card. OpenAI website, Aug 2025c. URL <https://cdn.openai.com/gpt-5-system-card.pdf>.
- Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. *arXiv preprint arXiv:2412.21139*, 2024.
- Jiayi Pan, Xiuyu Li, Long Lian, Charlie Victor Snell, Yifei Zhou, Adam Yala, Trevor Darrell, Kurt Keutzer, and Alane Suhr. Learning adaptive parallel reasoning with language models. In *Second Conference on Language Modeling*, 2025. URL <https://openreview.net/forum?id=YgwQ7sXPXU>.
- Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2024. URL <https://arxiv.org/abs/2412.15115>.
- ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, et al. Seed1. 5-thinking: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*, 2025.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.
- Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL <https://openreview.net/forum?id=OJd3ayDDoF>.

- Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL <https://openreview.net/forum?id=OJd3ayDDoF>.
- Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025.
- Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024.
- Chengxing Xie, Bowen Li, Chang Gao, He Du, Wai Lam, Difan Zou, and Kai Chen. SWE-fixer: Training open-source LLMs for effective and efficient GitHub issue resolution. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 1123–1139, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.62. URL <https://aclanthology.org/2025.findings-acl.62/>.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.
- John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024a.
- John Yang, Kilian Lieret, Carlos E Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*, 2025b.
- John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents, 2025c. URL <https://arxiv.org/abs/2504.21798>.
- Zonghan Yang, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. ReAct Meets ActRe: Autonomous Annotation of Agent Trajectories for Contrastive Self-Training. In *First Conference on Language Modeling*, 2024b. URL <https://openreview.net/forum?id=0VLBwQGwPA>.
- Liang Zeng, Yongcong Li, Yuzhen Xiao, Changshi Li, Chris Yuhao Liu, Rui Yan, Tianwen Wei, Jujie He, Xuchen Song, Yang Liu, et al. Skywork-swe: Unveiling data scaling laws for software engineering in llms. *arXiv preprint arXiv:2506.19290*, 2025.
- Linghao Zhang, Shilin He, Chaoyun Zhang, Yu Kang, Bowen Li, Chengxing Xie, Junhao Wang, Maoquan Wang, Yufan Huang, Shengyu Fu, et al. Swe-bench goes live! *arXiv preprint arXiv:2505.23419*, 2025.

APPENDIX

A DETAILS OF MID-TRAINING

We curate a mid-training data recipe with a focus on enhancing SWE capabilities. Central to this effort is the collection of pull request (PR) data from GitHub, which provides extensive coverage of real-world bug fixes, feature requests, and code enhancements. To ensure data quality, we apply two filters: (i) we only retain repositories that have accumulated at least five GitHub stars, thereby excluding sparsely maintained projects with limited community engagement; and (ii) we remove any repositories overlapping with the SWE-bench benchmark (Jimenez et al., 2023) to prevent potential data leakage. For each candidate repository, we query the GitHub API for all PRs with the state MERGED, while discarding those abandoned, superseded, or left under review. To preserve more context information, we also snapshot the entire codebase at the base commit before the first code change in the PR.

After data crawling, we incorporate two complementary forms for the natural code change data: (i) **natural diff patches** and (ii) **PR commit packs**. A natural diff patch consolidates all commits in a PR into the final code difference, typically expressed as SEARCH–REPLACE blocks. This format aligns with the Agentless paradigm, in which the model must directly output the final patch. In contrast, a commit pack captures the sequence of human-authored commits within a PR, where each commit message (textual reasoning) is paired with the corresponding code modification (action). This structure closely parallels the SWE-Agent setting, where intermediate reasoning steps are interleaved with actions. However, the distinction of the utilities for the two types of data is not absolute: commit messages in a PR commit pack can still inform the model’s knowledge and indirectly strengthen its reasoning ability in the Agentless setting.

Natural diff patches. The natural diff patches used in the mid-training data recipe are processed with the following rules:

- Incorporate the agentless prompt template (see Prompts 1,2,3,4; These four prompt templates are also used in the later stages, including cold-start, RL, and test-time self-play), and apply a loss mask to the prompt part. For the localization prompt, the response is set as the files modified in the ground-truth diff patch.
- If a related issue to the PR exists, use its content of the related issue; otherwise, use the PR title as the surrogate of the issue content.
- If a related issue to the PR exists, prepend the issue discussion at the beginning of the output in the code edit response. We aim to strengthen the model’s capability of code edit reasoning by leveraging the discussion contents.
- Discard PRs that include modifications to files other than `{.py, .md, .rst}`.
- For PRs containing `{.md, .rst}` file modifications, retain only the Python diffs and rewrite them into SEARCH–REPLACE blocks.
- Remove PRs involving file additions or deletions.
- For the code edits with only line insertions or deletions, preserve the original Git diff hunks as the SEARCH content in the SEARCH–REPLACE blocks.
- Ensure that no more than three Python files are modified per PR.
- Apply a filtering script to exclude PRs with non-`{.py, .md, .rst}` modifications, or PRs modifying more than three Python files.
- Further exclude PRs containing more than five SEARCH–REPLACE blocks.

A total of ~50B tokens for natural diff patches are obtained after applying these filtering rules.

```

1 Please look through the following GitHub problem description and
  Repository structure and provide a list of files that one would need
  to edit to fix the problem.
2
3 ### GitHub Problem Description ###
4 {related issue / PR title content}

```

```

810 5
811 6 ###
812 7
813 8 ### Repository Structure ###
814 9 {file structure induced by the repo snapshot}
815 10
816 11
817 12 ###
818 13
819 14 Please only provide the full path and return at most 5 files.
820 15 The returned files should be separated by new lines ordered by most to
821 16 least important and wrapped with ```
822 17 For example:
823 18 ```
824 19 file1.py
825 20 file2.py
826 21 ```

```

Listing 1: Agentless prompt template: Localization for BugFixer.

```

827 21 Please look through the following GitHub problem description and
828 22 Repository structure and provide a list of test files that should be
829 23 run after applying the patch to fix the issue.
830 24
831 25 ### GitHub Problem Description ###
832 26 {related issue / PR title content}
833 27
834 28 ###
835 29 ### Repository Structure ###
836 30 {file structure induced by the repo snapshot}
837 31
838 32 ###
839 33
840 34 Please only provide the full path and return at most 5 files.
841 35 The returned files should be separated by new lines ordered by most to
842 36 least important and wrapped with ```
843 37 For example:
844 38 ```
845 39 file1.py
846 40 file2.py
847 41 ```

```

Listing 2: Agentless prompt template: Localization for TestWriter.

```

849 41 We are currently solving the following issue within our repository.
850 42 Here is the issue text:
851 43 --- BEGIN ISSUE ---
852 44 {related issue / PR title content}
853 45 --- END ISSUE ---
854 46
855 47 Below are some code segments, each from a relevant file. One or more of
856 48 these files may contain bugs.
857 49
858 50 --- BEGIN FILE ---
859 51 ```
860 52 ### {filename1}
861 53 {content of filename1}
862 54
863 55 ### {filename2}
864 56 {content of filename2}
865 57 {...}

```

```

864 57
865 58   ```
866 59 --- END FILE ---
867 60
868 61 Please first localize the bug based on the issue statement, and then
869 62 generate *SEARCH/REPLACE* edits to fix the issue.
870 63
871 64 Every *SEARCH/REPLACE* edit must use this format:
872 65 1. The file path
873 66 2. The start of search block: <<<<<< SEARCH
874 67 3. A contiguous chunk of lines to search for in the existing source
875 68 code
876 69 4. The dividing line: =====
877 70 5. The lines to replace into the source code
878 71 6. The end of the replace block: >>>>>> REPLACE
879 72
880 73 Here is an example:
881 74
882 75 ```python
883 76 ### mathweb/flask/app.py
884 77 <<<<<< SEARCH
885 78 from flask import Flask
886 79 =====
887 80 import math
888 81 from flask import Flask
889 82 >>>>>> REPLACE
890 83 ```
891 84
892 85 Please note that the *SEARCH/REPLACE* edit REQUIRES PROPER INDENTATION.
893 86 If you would like to add the line ' print(x)', you must
894 87 fully write that out, with all those spaces before the code!
895 88 Wrap the *SEARCH/REPLACE* edit in blocks ```python...```.
896 89

```

Listing 3: Agentless prompt template: Code edit for BugFixer.

```

893 85 We are currently solving the following issue within our repository.
894 86 Here is the issue text:
895 87 --- BEGIN ISSUE ---
896 88 {related issue / PR title content}
897 89 --- END ISSUE ---
898 90
899 91 Below are some code segments, each from a relevant test file. One or
900 92 more of these files may be added some new tests which can reproduce
901 93 the issue.
902 94
903 95 --- BEGIN FILE ---
904 96 ```
905 97 ### {filename1}
906 98 {content of filename1}
907 99
908 100 ### {filename2}
909 101 {content of filename2}
910 102 {...}
911 103
912 104 --- END FILE ---
913 105
914 106 Please first localize some possible locations in those test files
915 107 within the repo, and then generate *SEARCH/REPLACE* edit updates to
916 108 the **test** files in the repo, so that the erroneous scenario
917 109 described in the problem is reproduced.
918 110
919 111 Every *SEARCH/REPLACE* edit must use this format:
920 112 1. The file path

```

```

918 109 2. The start of search block: <<<<<< SEARCH
919 110 3. A contiguous chunk of lines to search for in the existing source
920    code
921 111 4. The dividing line: =====
922 112 5. The lines to replace into the source code
923 113 6. The end of the replace block: >>>>>> REPLACE
924 114
925 115 Here is an example:
926 116
927 117 ```python
928 118 ### mathweb/flask/app.py
929 119 <<<<<< SEARCH
930 120 from flask import Flask
931 121 =====
932 122 import math
933 123 from flask import Flask
934 124
935 125 def test__rules__std_L060_raised() -> None:
936 126     try:
937 127         sql = "SELECT    IFNULL(NULL, 100),
938 128             NVL(NULL,100);"
939 129         result = lint(sql, rules=["L060"])
940 130         assert len(result) == 2
941 131     except:
942 132         print("Other issues")
943 133         return
944 134
945 135     try:
946 136         assert result[0]["description"] == "Use 'COALESCE' instead of '
947 137             IFNULL'."
948 138         assert result[1]["description"] == "Use 'COALESCE' instead of '
949 139             NVL'."
950 140         print("Issue resolved")
951 141     except AssertionError:
952 142         print("Issue reproduced")
953 143         return
954 144
955 145     return
956 146 >>>>>> REPLACE
957 147 ```
958 148 Please note that the *SEARCH/REPLACE* edit REQUIRES PROPER INDENTATION.
959    If you would like to add the line '         print(x)', you must
960    fully write that out, with all those spaces before the code!
961    Wrap the *SEARCH/REPLACE* edit in blocks ```python...```.

```

Listing 4: Agentless prompt template: Code edit for TestWriter.

PR commit packs. The PR commit packs used in the mid-training data recipe are processed with the following rules:

- Discard PRs that include modifications to files other than `{.py, .md, .rst}`.
- For `{.md, .rst}` file modifications, retain the “diff -git” signature but remove the actual content changes.
- Ensure that each PR modifies at most five Python files (with at least one required). PRs exceeding this limit are discarded.
- Apply a filtering script to exclude PRs containing non-`{.py, .md, .rst}` file modifications or those modifying more than five Python files.
- Filter out all of the developer signatures and GitHub IDs for ethics considerations.

A total of ~20B tokens for PR commit packs are obtained after applying these filtering rules.

In addition, we incorporate synthetic data to further enhance both the reasoning and agentic capabilities of the model. A key observation is that the ground-truth reward for the localization stage in the Agentless setting can be derived directly from the diff patch, since the set of files requiring modification is explicitly indicated.

Synthetic reasoning data. To improve reasoning quality, we perform a lightweight SFT of the Qwen-2.5-72B-Instruct model on 2,000 R1 trajectories. The resulting model is then used to generate large-scale rollouts for the localization stage of both BugFixer and TestWriter. We retain only the rollouts that achieve exactly correct file localizations. This procedure yields approximately $\sim 10\text{B}$ tokens of reasoning-intensive data dedicated to Agentless localization in the mid-training recipe.

Synthetic agentic interactions. To strengthen agentic capabilities, we simulate agent–environment interactions with a custom tool set designed to mimic file-system operations without execution. This design is motivated by practical constraints: while repository snapshots from GitHub are available, not all snapshots are equipped with an executable Docker environment. As a result, shell commands are disabled. Instead, we introduce synthetic tools that allow the agent to view file contents and perform keyword-based search for localization, which effectively reproduces the first stage of Agentless but in an agentic manner. The specification of this tool set is covered in the system prompt, which is then used to elicit agentic interaction rollouts from the Qwen-2.5-72B-Instruct model. The complete system prompt is provided in Prompt 5. We apply a loss mask only to the system prompt, and enable the model to simultaneously learn both actions and observations along the trajectory, inspired by Yang et al. (2024b). This approach integrates both policy and world modeling into mid training.

```

Your job is to look through the given GitHub problem description and
Repository structure, and edit updates to the files in the repo to
resolve the problem.
The job is divided into two stages:
+ In Stage 1, you should localize the files the files that you would
  need to edit to fix the problem.
+ In Stage 2, you should edit the updates to the repo.
Let's begin from Stage 1 to localize the bugs:

In Stage 1, besides reading the provided Repository structure, you can
use the following skills for exploration. The skills are to be
called in an environment wrapped by <execute> and </execute>, listed
in the form of python functions as below:

open_file(path: str, is_all | None = False, line_number: int | None =
1, context_lines: int | None = 100) -> None:
  Opens the file at the given path in the editor for exploration.
  By default, only the first 100 lines of the file are displayed. To
  open the entire file, set 'is_all' to 'True'.
  The 'context_lines' parameter determines the maximum number of
  lines to be displayed, with a cap of 100 lines. Use 'scroll_up'
  and 'scroll_down' to view more content up or down.
  If a 'line_number' is provided, the window will be moved to include
  that line.
  Note: When 'is_all' is set to 'True', the 'line_number' and '
  context_lines' parameters will not take effect, as the entire
  file will be opened and displayed without any line-specific
  focus or context limitation.
  Args:
  path: str: The path to the file to open. the full path of the
  filename should be provided.
  is_all: bool | None = False: If set to 'True', the entire file will
  be opened. Defaults to 'False'.
  line_number: int | None = 1: The line number to move to. Defaults
  to 1.
  context_lines: int | None = 100: Only shows this number of lines in
  the context window (usually from line 1), with line_number as
  the center (if possible). Defaults to 100.

goto_line(line_number: int) -> None:
  Moves the window to show the specified line number.

```

```

1026 171     Args:
1027 172     line_number: int: The line number to move to.
1028 173
1029 174 goto_class_or_func(class_or_func_name: str) -> None:
1030 175     Moves the window to show the specified class or function in the
1031 176     current open file.
1032 177     Args:
1033 178     class_or_func_name: str: The name of the given class, function, or
1034 179     method in a class to move to.
1035 180
1036 181 scroll_down() -> None:
1037 182     Moves the window down by 100 lines.
1038 183     Args:
1039 184     None
1040 185
1041 186 scroll_up() -> None:
1042 187     Moves the window up by 100 lines.
1043 188     Args:
1044 189     None
1045 190
1046 191 search_dir(search_term: str, dir_path: str | None) -> None:
1047 192     Searches for search_term in all files in dir. If dir is not
1048 193     provided, searches in the entire repository. Filename, fine-
1049 194     grained line number, and the relative class or function it is
1050 195     located in (if applied) will be shown for each found position.
1051 196     Args:
1052 197     search_term: str: The term to search for.
1053 198     dir_path: str: The path to the directory to search. Should be full
1054 199     path filename.
1055 200
1056 201 search_file(search_term: str, file_path: str | None = None) -> None:
1057 202     Searches for search_term in file. If file is not provided, searches
1058 203     in the current open file. Filename, fine-grained line number,
1059 204     and the relative class or function it is located in (if applied)
1060 205     will be shown for each found position.
1061 206     Args:
1062 207     search_term: str: The term to search for.
1063 208     file_path: str | None: The path to the file to search. Should be
1064 209     full path filename if provided.
1065 210
1066 211 find_file(file_name: str, dir_path: str | None) -> None:
1067 212     Finds all files with the given name in the specified directory. If
1068 213     dir is not provided, find in the entire repository.
1069 214     Args:
1070 215     file_name: str: The name of the file to find.
1071 216     dir_path: str: The path to the directory to search.
1072 217
1073 218 str_replace(path: str, old_str, new_str)
1074 219 old_str=[the old content to be replaced]
1075 220 new_str=[the new content after replacement]
1076 221 -> None:
1077 222     Replace the old content (old_str) in the file at the given path
1078 223     with the new content (new_str). This is the skill that you will
1079 224     be using to edit the updates.
1080 225     Args:
1081 226     path: str: The path to the file to be updated. The full path of the
1082 227     filename should be provided.
1083 228     old_str: str: The old content to be replaced. Note that this
1084 229     argument should be written in a new line starting with "old_str
1085 230     =", and the string content should not be quoted.
1086 231     new_str: str: The new content after replacement. Note that this
1087 232     argument should be written in a new line starting with "new_str
1088 233     =", and the string content should not be quoted.
1089 234
1090 235 Example:

```



```

1080 218     Assuming a call is shown as follows:
1081 219     ```
1082 220     str_replace("filename.py", old_str, new_str)
1083 221     old_str=      a
1084 222
1085 223     new_str=      b
1086 224     c
1087 225     ```
1088 226         Then it will function as replacing the '      a\n' string with the '
1089 227         b\nc ' string in the 'filename.py' file.
1090 228
1091 229 insert(path: str, insert_line: int, new_str)
1092 230 new_str=[the new content to be inserted]
1093 231 -> None:
1094 232     Insert the new content (new_str) in the file at the given path.
1095 233     When you want to add an entirely new class/function to the file,
1096 234     it would be better to use this method.
1097 235
1098 236 Args:
1099 237 path: str: The path to the file to be updated. The full path of the
1100 238 filename should be provided.
1101 239 insert_line: int: The Line number below which the new content is to
1102 240 be added. This Line number should be within the range of lines
1103 241 of the file: [0, Lines_of_the_File]. Specifically, when
1104 242 insert_line = 0, the added content starts from the top of the
1105 243 file.
1106 244 new_str: str: The new content to be inserted. Note that this
1107 245 argument should be written in a new line starting with "new_str
1108 246 =", and the string content should not be quoted.
1109 247
1110 248 Example:
1111 249 Assuming a call is shown as follows:
1112 250 ```
1113 251 insert("test_filename.py", 5, new_str)
1114 252 new_str=      def test_add():
1115 253     assert add(1, 2) == 3
1116 254 ```
1117 255     Then it will function as inserting the string '      def test_add():\
1118 256     n      assert add(1, 2) == 3' below the Line 5 of the '
1119 257     test_filename.py' file.
1120 258
1121 259 stop() -> None:
1122 260     Terminate the editing process.
1123 261     Args:
1124 262     None
1125 263
1126 264 NOTE:
1127 265 Responses should be concise.
1128 266 When exploring, you should attempt fewer things at a time: Include ONLY
1129 267 ONE <execute> per response, and use a SINGLE skill listed above
1130 268 within the <execute> environment. DO NOT use other python functions,
1131 269 as the environment does not support them.
1132 270 You should first reason in the verbal form, then use a skill with <
1133 271 execute> and </execute>.
1134 272 You should avoid apologies and thanks in the responses.
1135 273
1136 274 When you finish exploring and analyzing with the provided skills,
1137 275 please return at most 3 files with the full path only. Each full
1138 276 path should be placed in a single line, INSTEAD OF BROKEN WITH
1139 277 MULTIPLE LINES.
1140 278 The returned files should be separated by new lines ordered by most to
1141 279 least important, wrapped with ``` and NOTHING ELSE.
1142 280 An example for a full output:
1143 281 ```
1144 282 full_path_to_file1.py

```

```

1134 263 full_path_to_file2.py
1135 264 ```
1136 265
1137 266 Now Let's start!
1138 267
1139 268 ### GitHub Problem Description ###
1140 269
1141 270 {issue content}
1142 271
1143 272 ### Repository Structure ###
1144 273
1145 274 {file structure}
1146 275
1147 276 ###

```

Listing 5: A non-execution set of tools empowering the simulation of agentic interaction trajectories.

After completing the initial localization stage, the agent is guided into the code-editing phase through a follow-up instruction: “Now let’s move on to Stage 2 and edit the updates. Remember, you can still decide at any point whether a file actually requires modification.” We retain partial rollouts from Stage 1, provided that the localization results include at least one correct file.

In Stage 2, we first simulate the agent’s interaction by allowing it to open incorrectly localized files, and we artificially inject agentic reasoning patterns such as “I realize that I do not need to modify this file” after inspecting the file content. This procedure is designed to strengthen the self-reflection ability of the agent by exposing it to false-positive contexts regarding the issue to be solved.

Subsequently, we transcribe the ground-truth PR commit pack into trajectory form: each commit message is treated as the agent’s reasoning step, and each code update is represented as the corresponding action, expressed through the “str.replace” or “insert” tools. These interactions are appended to the trajectory, followed by a terminating “stop” call. Due to storage constraints on repository snapshots, this trajectory simulation is applied to only a subset of PRs. Overall, this process contributes approximately $\sim 10\text{B}$ tokens of agentic interaction data to the mid-training recipe. Future directions for scaling this component in the data recipe include leveraging the idea of environment scaling (Yang et al., 2025c).

Training. We perform mid-training using a standard next token prediction approach, initialized from the Qwen2.5-72B-Base (Qwen et al., 2024) model. We upsample the synthetic part of the data by a factor of 4 during mid-training, inspired by the practice in Grattafiori et al. (2024); Qwen et al. (2024); Gu et al. (2025). A global batch size of 256 with a maximum sequence length of 32K tokens is used, optimizing for long-context capabilities necessary for real-world software engineering tasks. The learning rate is set to $2\text{e-}5$, with a cosine decay schedule and a minimum learning rate of $2\text{e-}6$. The warm-up phase covers over approximately 3 billion tokens, followed by learning rate decay until approximately 150 billion tokens are processed. The training was conducted on 256 NVIDIA H100 GPUs and lasted for 213 hours.

B DOCKER ENVIRONMENTS

Table 3: The sources of the docker environments used in the development of Kimi-Dev.

Dataset Name	Dataset Link	Number of Dockers
SWE-Gym (Pan et al. (2024))	https://huggingface.co/datasets/SWE-Gym/SWE-Gym/	2,356
SWE-bench-extra (Badertdinov et al. (2024a))	https://huggingface.co/datasets/nebius/SWE-bench-extra/	3,846
R2E-Gym-Lite (Jain et al. (2025))	https://huggingface.co/datasets/R2E-Gym/R2E-Gym-Lite	3,671

Docker environment construction. To validate non-ground-truth patches generated by model roll-outs and expand our dataset, we required executable Docker environments. We combined publicly available datasets with custom-configured Docker environments (see Table. 3). Among them, SWE-Gym and R2E-Gym-Lite open-source their dockers that we can directly use. For datasets lacking Docker support (SWE-Bench-Extra), we implemented an automated configuration method:

1. Initialize a Docker environment with fixed dependencies.
2. Select Python version based on commit year.
3. Install dependencies via `requirements.txt` and `"pip install -e ."`.
4. Resolve `ModuleNotFoundError` errors during test execution.
5. Validate success if a `FAIL_TO_PASS` test transitions from failing (without `gt_patch`) to passing (with `gt_patch`).

Out of 6.38k SWE-bench-extra instances, 3,846 environments are successfully constructed and subsequently used for cold-start and RL training.

C MORE DETAILS OF RL TRAINING

C.1 RL TRAINING OBJECTIVE

For the RL algorithm, we adopt the policy optimization method proposed by Kimi k1.5 (Team et al., 2025), which has shown promising results on reasoning tasks in both math and code. Unlike GRPO (Shao et al., 2024), which eliminates the critic by using average rewards for advantage estimation, Kimi k1.5 (Team et al., 2025) adopts a simpler policy gradient approach. Specifically, we optimize the policy π_θ using the following objective:

$$\mathcal{J}_\theta = \mathbb{E}_{q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_\theta(\cdot|q)} \left\{ \sum_{i=1}^G \left[\left(R_i(q, o_i) - \text{mean}(\{R_j\}_{j=1}^G) - \tau \log \frac{\pi_\theta(o_i|q)}{\pi_{\theta_{ref}}(o_i|q)} \right) \right] \right\} \quad (2)$$

where q is a prompt sampled from the prompt data distribution $P(Q)$. $\{o_i\}_{i=1}^G$ are G responses generated by the current policy π_θ for prompt q . $R_i(q, o_i)$ is the reward for response o_i . The mean reward across the G responses is used as a baseline to center the reward (similar to advantage estimation). The final term penalizes divergence from a reference policy $\pi_{\theta_{ref}}$, with weight τ . This formulation avoids the need for a critic or value model by using group-wise relative rewards, and supports stable updates through KL regularization.

C.2 PROMPT SET SELECTION

In the main text, we introduce the adaptive prompt selection method for RL training. Specifically, we construct an initial prompt set of 1,200 problems by selecting those with `pass@16` > 0 from SWE-Gym (Pan et al., 2024), SWE-bench-extra (Badertdinov et al., 2025), and R2E-gym (Jain et al., 2025). Then, every 100 training steps, we expand the prompt set by adding 500 new problems. These additional problems are randomly sampled from the pool of problems for which the current model has `pass@16` = 0, thereby progressively increasing the difficulty.

C.3 RL EXPERIMENT ABLATION

Figure 9 shows the performance of the Qwen2.5-14B model in RL experiments, where both the BugFixer and the TestWriter exhibit clear scaling law behavior.

Furthermore, Figure 8 illustrates the effect of our proposed positive example reinforcement. We incorporated this improvement in the later stage of training and found that the RL of BugFixer

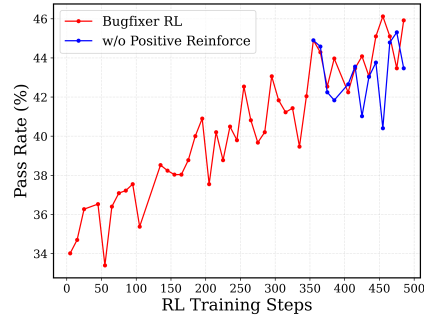


Figure 8: Ablation of positive example reinforcement during 72B Bugfixer RL.

achieved better performance compared to not using this trick. The main reason is that positive examples can reinforce the correct reasoning patterns, enabling the model to converge more quickly, especially when its exploration ability diminishes in the later stages. It is noticed that a similar trick was also applied in the development of the Seed1.5-thinking model (Seed et al., 2025).

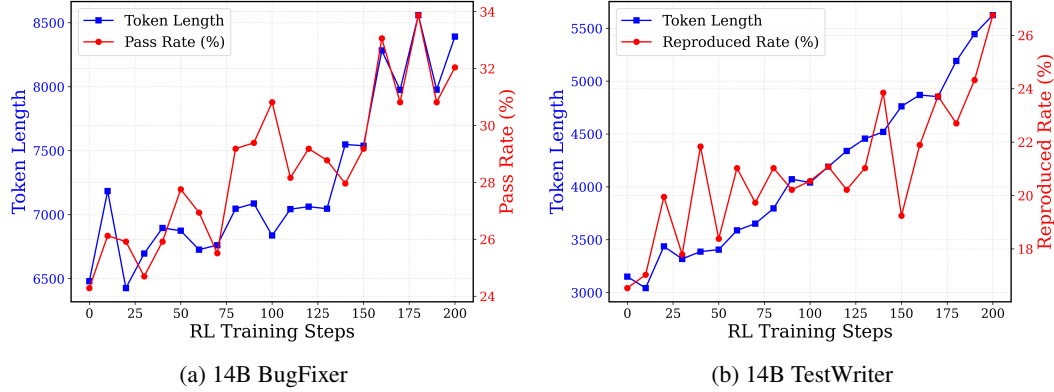


Figure 9: RL scaling experiments on Qwen2.5-14B model.

D AGENT INFRASTRUCTURE

We present a unified sandbox architecture designed for hybrid cloud cluster environments, leveraging Kubernetes orchestration to provide researchers with a flexible, secure, and highly scalable code execution platform. The architecture is accessible through a Python SDK and implements an ephemeral, use-and-destroy paradigm that ensures resource efficiency and security isolation. The system demonstrates exceptional scalability, supporting over 10,000 concurrent sandbox instances while maintaining performance stability. This infrastructure serves as a foundational platform for diverse computational scenarios, including automated code evaluation and RL experiments.

The system employs a sidecar container pattern to efficiently orchestrate and manage a heterogeneous collection of over 25,000 docker images sourced from multiple datasets. This architectural choice enables optimal resource utilization while maintaining isolation between different execution environments. Additionally, we have developed a proprietary Software Engineering (SWE) image-building pipeline that continuously expands the repository of supported images, thereby enhancing the system’s adaptability to diverse computational workloads and research requirements.

E ANALYSIS FOR SWE-AGENTS

E.1 STAGE ANNOTATION FOR SWE-AGENT TRAJECTORIES

In this section, we present how we use a frontier LLM to annotate the SWE-Agent stage to which each interaction turn within the trajectory rollout belongs. While we have briefly introduced the five stages suggested in the prompt of the SWE-Agent prompt in Section 4, we attach the excerpt in Listing 6 for greater clarity:

```

1289 277 ...
1290 278 Follow these steps to resolve the issue:
1291 279 1. As a first step, it might be a good idea to find and read code
1292 280    relevant to the <pr_description>
1293 281 2. Create a script to reproduce the error and execute it with 'python <
1294 282    filename.py>' using the bash tool, to confirm the error
1295 283 3. Edit the source code of the repo to resolve the issue
1296 284 4. Rerun your reproduce script and confirm that the error is fixed!
1297 285 5. Think about edgecases and make sure your fix handles them as well

```

...

Listing 6: The excerpt of the five-stage declaration in the SWE-Agent prompt.

It should be noted that the agent could flexibly transit across the five stages during its working process. For example, after Stage 4 when the agent rerun the test script, possibilities are that erroneous information remains, and this is when the agent goes back to Stage 3 to refine its code repair with reflection; Similar backtracing behavior could be observed from Stage 5 to Stage 3 as well, where the initial code repair has proven correct under the initial test script the agent composes in Stage 2, but fails some edge testcase the agent proposes in Stage 5.

To further analyze the BugFixer and the reflection skill prior, we need to realize which stage each turn along the SWE-Agent trajectory belongs to. As no strict boundaries or special prompt notes are set between each consecutive stage, we leverage an LLM for annotation. The annotation system prompt we set in kimi-k2-0711-preview is shown in Listing 7:

```
You are a professional inspector that can analyze the provided agentic
interaction trajectory.

The trajectory you are going to analyze is made by an agent that
interacts with a computer to solve tasks. This agent has access to
the following functions:

---- BEGIN FUNCTION #1: bash ----
Description: Execute a bash command in the terminal.

Parameters:
  (1) command (string, required): The bash command to execute. Can be
      empty to view additional logs when previous exit code is '-1'. Can
      be 'ctrl+c' to interrupt the currently running process.
---- END FUNCTION #1 ----

---- BEGIN FUNCTION #2: submit ----
Description: Finish the interaction when the task is complete OR if the
assistant cannot proceed further with the task.
No parameters are required for this function.
---- END FUNCTION #2 ----

---- BEGIN FUNCTION #3: str_replace_editor ----
Description: Custom editing tool for viewing, creating and editing
files
* State is persistent across command calls and discussions with the
user
* If 'path' is a file, 'view' displays the result of applying 'cat -n'.
  If 'path' is a directory, 'view' lists non-hidden files and
  directories up to 2 levels deep
* The 'create' command cannot be used if the specified 'path' already
  exists as a file
* If a 'command' generates a long output, it will be truncated and
  marked with '<response clipped>'
* The 'undo_edit' command will revert the last edit made to the file at
  'path'

Notes for using the 'str_replace' command:
* The 'old_str' parameter should match EXACTLY one or more consecutive
  lines from the original file. Be mindful of whitespaces!
* If the 'old_str' parameter is not unique in the file, the replacement
  will not be performed. Make sure to include enough context in '
  old_str' to make it unique
* The 'new_str' parameter should contain the edited lines that should
  replace the 'old_str'

Parameters:
```

```

1350 315 (1) command (string, required): The commands to run. Allowed options
1351 316 are: 'view', 'create', 'str_replace', 'insert', 'undo_edit'.
1352 317 Allowed values: ['view', 'create', 'str_replace', 'insert', 'undo_edit
1353 318 ']
1354 319 (2) path (string, required): Absolute path to file or directory, e.g.
1355 320 '/repo/file.py' or '/repo'.
1356 321 (3) file_text (string, optional): Required parameter of 'create'
1357 322 command, with the content of the file to be created.
1358 323 (4) old_str (string, optional): Required parameter of 'str_replace'
1359 324 command containing the string in 'path' to replace.
1360 325 (5) new_str (string, optional): Optional parameter of 'str_replace'
1361 326 command containing the new string (if not given, no string will be
1362 327 added). Required parameter of 'insert' command containing the
1363 328 string to insert.
1364 329 (6) insert_line (integer, optional): Required parameter of 'insert'
1365 330 command. The 'new_str' will be inserted AFTER the line '
1366 331 insert_line' of 'path'.
1367 332 (7) view_range (array, optional): Optional parameter of 'view'
1368 333 command when 'path' points to a file. If none is given, the full
1369 334 file is shown. If provided, the file will be shown in the
1370 335 indicated line number range, e.g. [11, 12] will show lines 11 and
1371 336 12. Indexing at 1 to start. Setting '[start_line, -1]' shows all
1372 337 lines from 'start_line' to the end of the file.
1373 338 ---- END FUNCTION #3 ----
1374 339
1375 340 The agent was instructed with the following:
1376 341
1377 342 * A python code repository has been uploaded in the directory /testbed.
1378 343 * Implement the necessary changes to the repository so that the
1379 344 requirements specified in the <pr_description> are met.
1380 345 * All changes to any of the test files described in the <pr_description
1381 346 > have already been taken care of. This means no need to modify the
1382 347 testing logic or any of the tests in any way.
1383 348 * Make the minimal changes to non-tests files in the /testbed directory
1384 349 to ensure the <pr_description> is satisfied.
1385 350
1386 351 The agent was suggested to follow the following steps to resolve the
1387 352 issue:
1388 353 1. As a first step, it might be a good idea to find and read code
1389 354 relevant to the <pr_description>
1390 355 2. Create a script to reproduce the error and execute it with 'python <
1391 356 filename.py>' using the bash tool, to confirm the error
1392 357 3. Edit the source code of the repo to resolve the issue
1393 358 4. Rerun your reproduce script and confirm that the error is fixed!
1394 359 5. Think about edgecases and make sure your fix handles them as well
1395 360 The agent was encouraged to think thoroughly, and it's fine if it's
1396 361 very long.
1397 362
1398 363 You are going to inspect this agent's interaction trajectory with a
1399 364 computer to solve the given task in the <pr_description>. One turn
1400 365 of interaction contains a pair of OBSERVATION and ACTION, where the
1401 366 OBSERVATION comes from the computer, and the ACTION is taken by the
1402 367 agent.
1403 368
1404 369 For each turn of interaction, determine which step (of the
1405 370 aforementioned five) this turn belongs to. Output a single number
1406 371 (1~5) ONLY in a separate line as your classification (DO NOT OUTPUT
1407 372 ANY OTHER WORDS THAN THE DIGIT).
1408 373
1409 374 You can think before make the inspection. When thinking, wrap your
1410 375 thought with <think> and </think>. Don't forget to output your final
1411 376 inspection after thinking.

```

Listing 7: The annotation prompt for SWE-Agent stages.

To provide a clearer understanding of the trajectory, we incorporate most of the tool descriptions and instructions from the SWE-Agent system prompt into the annotation system prompt. The annotation is conducted in a multi-round manner, leveraging the agent’s previous actions and observations, as well as the stage classifications of earlier turns, to better exploit contextual information. At the i -th round of annotation, the observation–action pair from turn i of the SWE-Agent trajectory is appended as input, and the annotator is expected to output the corresponding stage classification.

E.2 COMPARATIVE STUDY

Based on the automatic stage annotation in the above section, we present a comparative study by inspecting the performance on `sympy--sympy-20590` among the Kimi-Dev under Agentless, and each of the Base, MT, SFT, and RL priors with SWE-Agent adaptation.

The problem statement of `sympy--sympy-20590` is listed in Listing 8:

```
Symbol instances have __dict__ since 1.7?
In version 1.6.2 Symbol instances had no '__dict__' attribute
```python
>>> sympy.Symbol('s').__dict__

AttributeError Traceback (most recent call last)
<ipython-input-3-e2060d5eec73> in <module>
----> 1 sympy.Symbol('s').__dict__

AttributeError: 'Symbol' object has no attribute '__dict__'
>>> sympy.Symbol('s').__slots__
('name',)
```

This changes in 1.7 where 'sympy.Symbol('s').__dict__' now exists (and
returns an empty dict)
I may misinterpret this, but given the purpose of '__slots__', I assume
this is a bug, introduced because some parent class accidentally
stopped defining '__slots__'.
```

Listing 8: The problem statement of `sympy--sympy-20590`.

It is observed that the main difficulty in resolving the issue lies in the realization of the “*some parent class*” referenced in the problem. In fact, the hints text of this problem, which reflects the discussion of the developers under the original issue, reveals a much more in-depth investigation into the issue (Listing 9):

```
It seems that Basic now inherits 'DefaultPrinting' which I guess doesn't
have slots. I'm not sure if it's a good idea to add '__slots__' to
that class as it would then affect all subclasses.

...

Using slots can break multiple inheritance but only if the slots are
non-empty I guess. Maybe this means that any mixin should always
declare empty slots or it won't work properly with subclasses that
have slots...

I see that 'EvalfMixin' has '__slots__ = ()'.
I guess we should add empty slots to DefaultPrinting then.
```

Listing 9: The excerpted hints text of `sympy--sympy-20590`.

According to the discussion, it is clear that the code repair would be to “add empty slots to Default-Printing”, which naturally leads to the navigation towards the file related to the implementation of

the printer (`sympy/core/_print_helpers.py`, which is also the file updated by the ground-truth patch.) However, the `hints_text` information in the test set is *not* allowed to be used in the problem-solving process, which challenges the reasoner or the agent to figure out “the parent class that stopped defining `__slots__`” autonomously.

We first examine Kimi-Dev under Agentless. None of the 40 runs succeeded in producing the correct file localization. In most cases, the updates are made to `sympy/core/symbol.py`, which is a plausible choice since the reported problem is triggered by `sympy.Symbol('s')`, and `symbol.py` should contain the definition of the `Symbol` class. This line of reasoning, however, diverges from the correct solution. A small number of runs attempt alternative localizations, such as `sympy/core/{basic.py, expr.py}`, but these are also incorrect.

The limitation of the Agentless framework becomes more obvious when the focus is shifted to the inspection over the SWE-Agent trajectories. For all of the Base, MT, SFT, and RL priors with SWE-Agent adaptation, the resulting trajectories always contain a test-reproducing phase (marked as Stage “2” by the annotator), with the generated script reads like:

```
from sympy import Symbol

# Try to access __dict__ on a Symbol instance
s = Symbol('s')
print("Symbol.__slots__:", s.__slots__)
try:
    print("Symbol.__dict__:", s.__dict__)
except AttributeError as e:
    print("Error:", e)
```

Listing 10: An exemplar reproducing script in Stage 2 within the SWE-Agent trajectories in `sympy__sympy-20590`.

It is trivial for an agent to generate a reproducing script of this form, as it directly transcribes the case described in the problem statement. Executing the script provides an unambiguous signal of whether the issue is resolved: the attribute `s.__dict__` should no longer exist, and the expected `Error:` message should appear in the standard error output. Indeed, observing the initial completion of Stage 3 across all trajectories reveals that none of the agents succeeded on their first attempt. However, the incomplete status is consistently exposed in Stage 4, when the script is rerun and `s.__dict__` still persists. This progression highlights the advantage of the SWE-Agent framework over the Agentless paradigm, as the agent is afforded additional opportunities for iterative refinement rather than being limited to a single attempt.

We continue to analyze the behavior of the agents adapted from the Base, MT, and SFT priors. Due to the difficulty of the issue, it is non-trivial to correctly localize the error through their agentic reasoning: the interactions show that the agents were switching among several file candidates, but failed to reach the correct one. The only exception comes from the agent adapted from the RL prior, which marks its unique success on this instance through accurate agentic reasoning. We attach the intermediate process from the initial Stage-3 moment to the Stage-3 revisit (refining the initial code repair) within the successful trajectory, tagged by the stage annotator:

```
...

OBSERVATION:
Symbol.__slots__: ('name',)
Symbol.__dict__: {}

ACTION:
Hmm, it seems like the issue is still there. Let's check if there are
any other classes in the hierarchy that might be missing '__slots__':

<function=bash>
<parameter=command>python -c "from sympy import Symbol; s = Symbol('s')
; for cls in type(s).__mro__: print(cls.__name__, 'has __slots__:',
hasattr(cls, '__slots__'))"</parameter>
```

```

1512 389 </function>
1513 390
1514 391 OBSERVATION:
1515 392 File "<string>", line 1
1516 393     from sympy import Symbol; s = Symbol('s'); for cls in type(s).
1517 394         __mro__: print(cls.__name__, 'has __slots__:', hasattr(cls, '
1518 395         __slots__'))
1519 396
1520 397 SyntaxError: invalid syntax
1521 398
1522 399 ACTION:
1523 400 Let's try a different approach:
1524 401
1525 402 <function=str_replace_editor>
1526 403 <parameter=command>create</parameter>
1527 404 <parameter=path>/testbed/check_slots.py</parameter>
1528 405 <parameter=file_text>
1529 406 from sympy import Symbol
1530 407
1531 408 s = Symbol('s')
1532 409 for cls in type(s).__mro__:
1533 410     print(f"{cls.__name__:15} has __slots__: {hasattr(cls, '__slots__')}
1534 411     ")
1535 412     if hasattr(cls, '__slots__'):
1536 413         print(f"{' '*16}__slots__ value: {cls.__slots__}")
1537 414 </parameter>
1538 415 </function>
1539 416
1540 417 OBSERVATION:
1541 418 File created successfully at: /testbed/check_slots.py
1542 419
1543 420 ACTION:
1544 421 <function=bash>
1545 422 <parameter=command>python /testbed/check_slots.py</parameter>
1546 423 </function>
1547 424
1548 425 OBSERVATION:
1549 426 Symbol          has __slots__: True
1550 427         __slots__ value: ('name',)
1551 428 AtomicExpr      has __slots__: True
1552 429         __slots__ value: ()
1553 430 Atom            has __slots__: True
1554 431         __slots__ value: ()
1555 432 Expr            has __slots__: True
1556 433         __slots__ value: ()
1557 434 Boolean         has __slots__: True
1558 435         __slots__ value: ()
1559 436 Basic           has __slots__: True
1560 437         __slots__ value: ('_mhash', '_args', '_assumptions')
1561 438 Printable       has __slots__: False
1562 439 EvalfMixin      has __slots__: True
1563 440         __slots__ value: ()
1564 441 object          has __slots__: False
1565 442
1566 443 ACTION:
1567 444 Now I see the issue! 'Printable' doesn't have '__slots__' defined. Let's
1568 445 check the 'Printable' class:
1569 446
1570 447 <function=str_replace_editor>
1571 448 <parameter=command>view</parameter>
1572 449 <parameter=path>/testbed/sympy/core/_print_helpers.py</parameter>
1573 450 <parameter=view_range>[1, 30]</parameter>
1574 451 </function>
1575 452
1576 453 OBSERVATION:

```

```

1566 450 Here's the result of running `cat -n` on /testbed/sympy/core/
1567      _print_helpers.py:
1568 451      1 """
1569 452      2 Base class to provide str and repr hooks that `init_printing`
1570      can overwrite.
1571 453      3
1572 454      4 This is exposed publicly in the `printing.defaults` module,
1573 455      5 but cannot be defined there without causing circular imports.
1574 456      6 """
1575 457      7
1576 458      8 class Printable:
1577 459      9     """
1578 460      10     The default implementation of printing for SymPy classes.
1579 461      11
1580 462      12     This implements a hack that allows us to print elements of
1581 463      13     built-in Python containers in a readable way. Natively Python uses
1582 464      14     ``repr()`` even if ``str()`` was explicitly requested. Mix in this
1583 465      15     trait into a class to get proper default printing.
1584 466      16
1585 467      17     This also adds support for LaTeX printing in jupyter
1586 468      18     notebooks.
1587 469      19
1588 470      20     # Note, we always use the default ordering (lex) in __str__
1589 471      and __repr__,
1590 472      21     # regardless of the global setting. See issue 5487.
1591 473      22     def __str__(self):
1592 474      23         from sympy.printing.str import sstr
1593 475      24         return sstr(self, order=None)
1594 476      25
1595 477      26     __repr__ = __str__
1596 478      27
1597 479      28     def _repr_disabled(self):
1598 480      29         """
1599 481      30         No-op repr function used to disable jupyter display
1600 482      hooks.
1601 483
1602 484 ACTION:
1603 485 We need to add `__slots__ = ()` to the `Printable` class as well:
1604 486
1605 487 <function=str_replace_editor>
1606 488 <parameter=command>str_replace</parameter>
1607 489 <parameter=path>/testbed/sympy/core/_print_helpers.py</parameter>
1608 490 <parameter=old_str>
1609 491 class Printable:
1610 492     """
1611 493     The default implementation of printing for SymPy classes.
1612 494
1613 495     This implements a hack that allows us to print elements of built-in
1614 496     Python containers in a readable way. Natively Python uses ``repr()``
1615 497     even if ``str()`` was explicitly requested. Mix in this trait into
1616 498     a class to get proper default printing.
1617 499
1618 500     This also adds support for LaTeX printing in jupyter notebooks.
1619 501     """
1620 502 </parameter>
1621 503 <parameter=new_str>
1622 504 class Printable:
1623 505     """
1624 506     The default implementation of printing for SymPy classes.

```

```

1620 506 This implements a hack that allows us to print elements of built-in
1621 507 Python containers in a readable way. Natively Python uses ``repr()
1622 ``
1623 508 even if ``str()`` was explicitly requested. Mix in this trait into
1624 509 a class to get proper default printing.
1625 510
1626 511 This also adds support for LaTeX printing in jupyter notebooks.
1627 512 """
1628 513
1629 514 __slots__ = ()
1629 515 </parameter>
1630 516 </function>
1631 517
1632 518 ...

```

Listing 11: The intermediate reflection process between the two Stage-3 moment of the successful trajectory in `sympy--sympy-20590`.

Listing 11 shows the advanced reflection capability through slot checking along the entire class hierarchy, which successfully leads to the observation of the erroneous `Printable`. The following behavior from the agent includes the routine of test rerun, and the observation of passing the test script (Listing 10). This chain of actions recovers and resonates with the developer discussions in `hints_text` (Listing 9), showing the superior reflection skill in the RL prior over other priors.

E.3 PATTERN COMPARISON BETWEEN SFT WITH EXPERT DATA AND END-TO-END RL

In this section, we compare the patterns of the interaction trajectories of the RL prior that undergoes (i) 1.5×2^{28} SWE-Agent expert SFT tokens (later referred as “heavy SFT”) and (ii) 2^{21} SWE-Agent expert SFT tokens + end-to-end SWE-Agent RL (later referred as “light SFT + end-to-end RL”) as a qualitative study for Figure 7.

We select an instance solved by both methods, `matplotlib--matplotlib-13989`, as case study. We attach the actions along the two interaction trajectories for comparison.

The action patterns of the heavy SFT model read as Listing 12:

```

1653 519 ACTION 0:
1654 520 I'll help you implement the necessary changes to fix the issue where `
1655 521 hist()` doesn't respect `range=...` when `density=True`. Let's
1656 522 follow the steps:
1657 523
1658 524 1. First, let's find and read the relevant code:
1659 525
1660 526 <function=bash>
1661 527 <parameter=command>find /testbed -type f -name "*.py" | grep -v "test_
1662 528 "</parameter>
1663 529 </function>
1664 530
1665 531 ACTION 1:
1666 532 Let's look for the histogram implementation since that's what we need
1667 533 to fix. The PR mentions that the bug is in the `hist()` function
1668 534 when `density=True` and `range=...` is specified.
1669 535
1670 536 <function=bash>
1671 537 <parameter=command>grep -r "def hist" --include="*.py" /testbed | grep
1672 538 -v "test_"</parameter>
1673 539 </function>
1674 540
1675 541 ACTION 2:
1676 542 The main histogram implementation appears to be in `/testbed/lib/
1677 543 matplotlib/axes/_axes.py`. Let's look at that file:

```

```

1674 538 <function=str_replace_editor>
1675 539 <parameter=command>view</parameter>
1676 540 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1677 541 </function>
1678 542
1679 543 ACTION 3:
1680 544 Let me look at the histogram implementation specifically:
1681 545
1682 546 <function=str_replace_editor>
1683 547 <parameter=command>view</parameter>
1684 548 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1685 549 <parameter=view_range>[6365, 6899]</parameter>
1686 550 </function>
1687 551
1688 552 ACTION 4:
1689 553 Let me also look at the 'histogram_bin_edges' function since it's used
1690 554 in the code:
1691 555
1692 556 <function=str_replace_editor>
1693 557 <parameter=command>view</parameter>
1694 558 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1695 559 <parameter=view_range>[40, 60]</parameter>
1696 560 </function>
1697 561
1698 562 ACTION 5:
1699 563 2. Let's create a script to reproduce the error:
1700 564
1701 565 <function=str_replace_editor>
1702 566 <parameter=command>create</parameter>
1703 567 <parameter=path>/testbed/reproduce.py</parameter>
1704 568 ...
1705 569
1706 570 ACTION 6:
1707 571 Let's run it to confirm the issue:
1708 572
1709 573 <function=bash>
1710 574 <parameter=command>python /testbed/reproduce.py</parameter>
1711 575 </function>
1712 576
1713 577 ACTION 7:
1714 578 The issue is confirmed: when 'density=True' is specified, the 'range
1715 579 =(0, 1)' parameter is not respected. The bins should start at 0 and
1716 580 end at 1, but they don't.
1717 581
1718 582 3. Let's fix the issue. Looking at the code, the problem is in the '
1719 583 hist' method and specifically around line 6689-6690:
1720 584
1721 585 ```python
1722 586 if density and not stacked:
1723 587     hist_kwargs = dict(density=density)
1724 588 ```
1725 589
1726 590 The issue is that when 'density=True', the 'range' parameter gets lost
1727 591 because we're not including it in the 'hist_kwargs'. Let's modify
1728 592 the code to fix this:
1729 593
1730 594 <function=str_replace_editor>
1731 595 <parameter=command>str_replace</parameter>
1732 596 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1733 597 ...
1734 598
1735 599 ACTION 8:
1736 600 4. Let's run the reproduce script again to confirm the fix:
1737 601
1738 602 <function=bash>

```



```

1728 597 <parameter=command>python /testbed/reproduce.py</parameter>
1729 598 </function>
1730 599
1731 600 ACTION 9:
1732 601 Great! The fix works. Now the bins start at 0 and end at 1 as expected.
1733 602
1734 603 5. Let's think about edge cases:
1735 604
1736 605 1. When 'range=None' - This should use the min/max of the data, which
1737 606 is already handled correctly by the existing code
1738 607 2. When 'range' is specified but 'density=False' - The range should be
1739 608 respected, which it already is
1740 609 3. When 'range' is specified and 'density=True' - The range should be
1741 610 respected, which our fix now ensures
1742 611 4. When 'bins' is a sequence - The range should be ignored, which is
1743 612 already handled by the code
1744 613 5. When 'range' includes values outside the data range - The histogram
1745 614 should include empty bins for the range outside the data, which is
1746 615 already handled by numpy's histogram function
1747 616
1748 617 Let me test some of these edge cases to be sure:
1749 618
1750 619 <function=str_replace_editor>
1751 620 <parameter=command>str_replace</parameter>
1752 621 <parameter=path>/testbed/reproduce.py</parameter>
1753 622 ...
1754 623
1755 624 ACTION 10:
1756 625 <function=bash>
1757 626 <parameter=command>python /testbed/reproduce.py</parameter>
1758 627 </function>
1759 628
1760 629 ACTION 11:
1761 630 All test cases look good:
1762 631
1763 632 1. When 'range=(0, 1)' and 'density=True', the bins start at 0 and end
1764 633 at 1
1765 634 2. When 'range=None' and 'density=True', the bins use the min/max of
1766 635 the data
1767 636 3. When 'range=(0, 1)' and 'density=False', the bins start at 0 and end
1768 637 at 1
1769 638 4. When 'bins' is a sequence, the range parameter is ignored and the
1770 639 sequence is used directly
1771 640 5. When 'range=(-0.5, 1.5)', the bins start at -0.5 and end at 1.5
1772 641
1773 642 The fix looks good and handles all edge cases correctly. Let me submit
1774 643 the changes:
1775 644
1776 645 <function=submit>
1777 646 </function>

```

Listing 12: The interaction made by the RL prior with 1.5×2^{28} SWE-Agent SFT tokens on matplotlib-matplotlib-13989.

The action patterns of the light SFT + end-to-end RL model read as Listing 13:

```

1775 636 ACTION 0:
1776 637 Understood. Let's tackle this problem step by step.
1777 638
1778 639 First, I need to find the relevant code in the /testbed directory. The
1779 640 user mentioned that the code is in /testbed, so I'll start by
1780 641 listing the files there.
1781 642
1782 643 <function=str_replace_editor>
1783 644 <parameter=command>view</parameter>

```

```

1782 643 <parameter=path>/testbed</parameter>
1783 644 </function>
1784 645
1785 646 ACTION 1:
1786 647 Hmm, the /testbed directory has a lot of files, but I need to focus on
1787 the code related to the hist function. The hist function is part of
1788 matplotlib.pyplot, so I should look for the pyplot module in the /
1789 testbed/lib/matplotlib directory.
1790 648
1791 649 <function=str_replace_editor>
1792 650 <parameter=command>view</parameter>
1793 651 <parameter=path>/testbed/lib/matplotlib/pyplot.py</parameter>
1794 652 </function>
1795 653
1796 654 ACTION 2:
1797 655 Okay, I'll look at the hist function in pyplot.py.
1798 656
1799 657 <function=str_replace_editor>
1800 658 <parameter=command>view</parameter>
1801 659 <parameter=path>/testbed/lib/matplotlib/pyplot.py</parameter>
1802 660 <parameter=view_range>[2610, 2625]</parameter>
1803 661 </function>
1804 662
1805 663 ACTION 3:
1806 664 The hist function in pyplot.py simply calls the hist method of the
1807 665 current axes. So the actual implementation is in the Axes class.
1808 666
1809 667 I'll look for the Axes.hist method in the /testbed/lib/matplotlib/axes
1810 668 directory.
1811 669
1812 670 <function=str_replace_editor>
1813 671 <parameter=command>view</parameter>
1814 672 <parameter=path>/testbed/lib/matplotlib/axes</parameter>
1815 673 </function>
1816 674
1817 675 ACTION 4:
1818 676 The Axes.hist method is likely in _axes.py, which is where most of the
1819 677 plotting methods are defined.
1820 678
1821 679 <function=str_replace_editor>
1822 680 <parameter=command>view</parameter>
1823 681 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1824 682 </function>
1825 683
1826 684 ACTION 5:
1827 685 I'll look at the hist method in _axes.py.
1828 686
1829 687 <function=str_replace_editor>
1830 688 <parameter=command>view</parameter>
1831 689 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1832 690 <parameter=view_range>[6365, 6400]</parameter>
1833 691 </function>
1834 692
1835 693 ACTION 6:
1836 694 I'll look at more of the hist method to see where the range is handled.
1837 695
1838 696 <function=str_replace_editor>
1839 697 <parameter=command>view</parameter>
1840 698 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1841 699 <parameter=view_range>[6400, 6450]</parameter>
1842 700 </function>
1843 701
1844 702 ACTION 7:
1845 703 I'll look at the code where the bins are calculated.

```

```

1836 702 <function=str_replace_editor>
1837 703 <parameter=command>view</parameter>
1838 704 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1839 705 <parameter=view_range>[6450, 6500]</parameter>
1840 706 </function>
1841 707
1842 708 ACTION 8:
1843 709 I'll look at the code where the bins are calculated.
1844 710
1845 711 <function=str_replace_editor>
1846 712 <parameter=command>view</parameter>
1847 713 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1848 714 <parameter=view_range>[6500, 6550]</parameter>
1849 715 </function>
1850 716
1851 717 ACTION 9:
1852 718 I'll look at the code where the bins are calculated.
1853 719
1854 720 <function=str_replace_editor>
1855 721 <parameter=command>view</parameter>
1856 722 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1857 723 <parameter=view_range>[6550, 6600]</parameter>
1858 724 </function>
1859 725
1860 726 ACTION 10:
1861 727 I'll look at the code where the bins are calculated.
1862 728
1863 729 <function=str_replace_editor>
1864 730 <parameter=command>view</parameter>
1865 731 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1866 732 <parameter=view_range>[6600, 6650]</parameter>
1867 733 </function>
1868 734
1869 735 ACTION 11:
1870 736 I'll look at the code where the bins are calculated.
1871 737
1872 738 <function=str_replace_editor>
1873 739 <parameter=command>view</parameter>
1874 740 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1875 741 <parameter=view_range>[6650, 6700]</parameter>
1876 742 </function>
1877 743
1878 744 ACTION 12:
1879 745 Ah, here's the key part. The code calculates the bins using np.
1880 746 histogram, and if the range is specified, it's passed to
1881 747 histogram_bin_edges. But when density is True, the code may be
1882 748 overriding the range.
1883 749
1884 750 Wait, in the code:
1885 751
1886 752 if not input_empty and len(x) > 1:
1887 753     if weights is not None:
1888 754         _w = np.concatenate(w)
1889 755     else:
1890 756         _w = None
1891 757
1892 758     bins = histogram_bin_edges(np.concatenate(x),
1893 759                               bins, bin_range, _w)
1894 760 else:
1895 761     hist_kwargs['range'] = bin_range
1896 762
1897 763 But when density is True, the code may be using the default range of
1898 764 the data, not the specified range.
1899 765
1900 766 Wait, no. The code for calculating the bins is:

```

```

1890 763
1891 764 if bins is None:
1892 765     bins = rcParams['hist.bins']
1893 766
1894 767 if bin_range is not None:
1895 768     bin_range = self.convert_xunits(bin_range)
1896 769
1897 770 if bin_range is None:
1898 771     xmin = np.inf
1899 772     xmax = -np.inf
1900 773     for xi in x:
1901 774         if len(xi):
1902 775             xmin = min(xmin, np.nanmin(xi))
1903 776             xmax = max(xmax, np.nanmax(xi))
1904 777     if not np.isnan([xmin, xmax]).any() and not (xmin > xmax):
1905 778         bin_range = (xmin, xmax)
1906 779
1907 780 # If bins are not specified either explicitly or via range,
1908 781 # we need to figure out the range required for all datasets,
1909 782 # and supply that to np.histogram.
1910 783 if not input_empty and len(x) > 1:
1911 784     if weights is not None:
1912 785         _w = np.concatenate(w)
1913 786     else:
1914 787         _w = None
1915 788
1916 789     bins = histogram_bin_edges(np.concatenate(x),
1917 790                               bins, bin_range, _w)
1918 791 else:
1919 792     hist_kwargs['range'] = bin_range
1920 793
1921 794 density = bool(density) or bool(normed)
1922 795 if density and not stacked:
1923 796     hist_kwargs = dict(density=density)
1924 797
1925 798 # List to store all the top coordinates of the histograms
1926 799 tops = []
1927 800 mlast = None
1928 801 # Loop through datasets
1929 802 for i in range(nx):
1930 803     # this will automatically overwrite bins,
1931 804     # so that each histogram uses the same bins
1932 805     m, bins = np.histogram(x[i], bins, weights=w[i], **hist_kwargs)
1933 806     m = m.astype(float) # causes problems later if it's an int
1934 807     if mlast is None:
1935 808
1936 809 But when density is True, the code passes density=True to np.histogram.
1937 810 However, when density is True, np.histogram normalizes the counts,
1938 811 but the bin edges are still determined by the range.
1939 812
1940 813 Wait, but the problem is that when density=True, the range is not
1941 814 respected. The user's code is:
1942 815
1943 816 _, bins, _ = plt.hist(np.random.rand(10), "auto", range=(0, 1), density
1944 817                       =True)
1945 818 print(bins)
1946 819
1947 820 and the output is:
1948 821
1949 822 [0.00331535 0.18930174 0.37528813 0.56127453 0.74726092 0.93324731]
1950 823
1951 824 which is not from 0 to 1. But when density=False, the range is
1952 825 respected.

```

1944 822 This suggests that when density=True, the code is not using the
1945 823 specified range for the bins.

1946 823 But why?

1947 824 Let me look at the code again.

1948 825

1949 826 When the code calculates the bins:

1950 827

1951 828

1952 829

1953 830 if not input_empty and len(x) > 1:

1954 831 if weights is not None:

1955 832 _w = np.concatenate(w)

1956 833 else:

1957 834 _w = None

1958 835

1959 836 bins = histogram_bin_edges(np.concatenate(x),

1960 837 bins, bin_range, _w)

1961 838 else:

1962 839 hist_kwargs['range'] = bin_range

1963 840

1964 841 But when len(x) is 1 (as in the user's code), the code enters the else

1965 842 clause and sets hist_kwargs['range'] = bin_range.

1966 843 Then, in the loop:

1967 844

1968 845 for i in range(nx):

1969 846 m, bins = np.histogram(x[i], bins, weights=w[i], **hist_kwargs)

1970 847

1971 848 But wait, if bins is None, then np.histogram will use the range. But if

1972 849 bins is a string (like 'auto'), then np.histogram will calculate

1973 850 the bins based on the data and the range.

1974 851

1975 852 Wait, in the user's code, bins is 'auto' and range is (0, 1). When

1976 853 density=True, the code should calculate the bins based on the range

1977 854 (0, 1).

1978 855

1979 856 But in the code, when len(x) is 1, the code sets hist_kwargs['range'] =

1980 857 bin_range. Then, in the loop, the code calls np.histogram(x[i],

1981 858 bins, weights=w[i], **hist_kwargs).

1982 859

1983 860 But if bins is a string (like 'auto'), then np.histogram will calculate

1984 861 the bins based on the data and the range. So the range should be

1985 862 respected.

1986 863

1987 864 But in the user's code, the bins are not from 0 to 1. This suggests

1988 865 that the code is not passing the range to np.histogram when density=

1989 866 True.

1990 867

1991 868 But why?

1992 869

1993 870 This is puzzling. Let me try to reproduce the error.

1994 871

1995 872 I'll create a script in the /testbed directory.

1996 873

1997 874

1998 875

1999 876

2000 877

2001 878

2002 879

2003 880

2004 881

2005 882

2006 883

2007 884

2008 885

2009 886

2010 887

2011 888

2012 889

2013 890

2014 891

2015 892

2016 893

2017 894

2018 895

2019 896

2020 897

2021 898

2022 899

2023 900

2024 901

2025 902

2026 903

2027 904

2028 905

2029 906

2030 907

2031 908

2032 909

2033 910

2034 911

2035 912

2036 913

2037 914

2038 915

2039 916

2040 917

2041 918

2042 919

2043 920

2044 921

2045 922

2046 923

2047 924

2048 925

2049 926

2050 927

2051 928

2052 929

2053 930

2054 931

2055 932

2056 933

2057 934

2058 935

2059 936

2060 937

2061 938

2062 939

2063 940

2064 941

2065 942

2066 943

2067 944

2068 945

2069 946

2070 947

2071 948

2072 949

2073 950

2074 951

2075 952

2076 953

2077 954

2078 955

2079 956

2080 957

2081 958

2082 959

2083 960

2084 961

2085 962

2086 963

2087 964

2088 965

2089 966

2090 967

2091 968

2092 969

2093 970

2094 971

2095 972

2096 973

2097 974

2098 975

2099 976

2100 977

2101 978

2102 979

2103 980

2104 981

2105 982

2106 983

2107 984

2108 985

2109 986

2110 987

2111 988

2112 989

2113 990

2114 991

2115 992

2116 993

2117 994

2118 995

2119 996

2120 997

2121 998

2122 999

2123 1000

2124 1001

2125 1002

2126 1003

2127 1004

2128 1005

2129 1006

2130 1007

2131 1008

2132 1009

2133 1010

2134 1011

2135 1012

2136 1013

2137 1014

2138 1015

2139 1016

2140 1017

2141 1018

2142 1019

2143 1020

2144 1021

2145 1022

2146 1023

2147 1024

2148 1025

2149 1026

2150 1027

2151 1028

2152 1029

2153 1030

2154 1031

2155 1032

2156 1033

2157 1034

2158 1035

2159 1036

2160 1037

2161 1038

2162 1039

2163 1040

2164 1041

2165 1042

2166 1043

2167 1044

2168 1045

2169 1046

2170 1047

2171 1048

2172 1049

2173 1050

2174 1051

2175 1052

2176 1053

2177 1054

2178 1055

2179 1056

2180 1057

2181 1058

2182 1059

2183 1060

2184 1061

2185 1062

2186 1063

2187 1064

2188 1065

2189 1066

2190 1067

2191 1068

2192 1069

2193 1070

2194 1071

2195 1072

2196 1073

2197 1074

2198 1075

2199 1076

2200 1077

2201 1078

2202 1079

2203 1080

2204 1081

2205 1082

2206 1083

2207 1084

2208 1085

2209 1086

2210 1087

2211 1088

2212 1089

2213 1090

2214 1091

2215 1092

2216 1093

2217 1094

2218 1095

2219 1096

2220 1097

2221 1098

2222 1099

2223 1100

2224 1101

2225 1102

2226 1103

2227 1104

2228 1105

2229 1106

2230 1107

2231 1108

2232 1109

2233 1110

2234 1111

2235 1112

2236 1113

2237 1114

2238 1115

2239 1116

2240 1117

2241 1118

2242 1119

2243 1120

2244 1121

2245 1122

2246 1123

2247 1124

2248 1125

2249 1126

2250 1127

2251 1128

2252 1129

2253 1130

2254 1131

2255 1132

2256 1133

2257 1134

2258 1135

2259 1136

2260 1137

2261 1138

2262 1139

2263 1140

2264 1141

2265 1142

2266 1143

2267 1144

2268 1145

2269 1146

2270 1147

2271 1148

2272 1149

2273 1150

2274 1151

2275 1152

2276 1153

2277 1154

2278 1155

2279 1156

2280 1157

2281 1158

2282 1159

2283 1160

2284 1161

2285 1162

2286 1163

2287 1164

2288 1165

2289 1166

2290 1167

2291 1168

2292 1169

2293 1170

2294 1171

2295 1172

2296 1173

2297 1174

2298 1175

2299 1176

2300 1177

2301 1178

2302 1179

2303 1180

2304 1181

2305 1182

2306 1183

2307 1184

2308 1185

2309 1186

2310 1187

2311 1188

2312 1189

2313 1190

2314 1191

2315 1192

2316 1193

2317 1194

2318 1195

2319 1196

2320 1197

2321 1198

2322 1199

2323 1200

2324 1201

2325 1202

2326 1203

2327 1204

2328 1205

2329 1206

2330 1207

2331 1208

2332 1209

2333 1210

2334 1211

2335 1212

2336 1213

2337 1214

2338 1215

2339 1216

2340 1217

2341 1218

2342 1219

2343 1220

2344 1221

2345 1222

2346 1223

2347 1224

2348 1225

2349 1226

2350 1227

2351 1228

2352 1229

2353 1230

2354 1231

2355 1232

2356 1233

2357 1234

2358 1235

2359 1236

2360 1237

2361 1238

2362 1239

2363 1240

2364 1241

2365 1242

2366 1243

2367 1244

2368 1245

2369 1246

2370 1247

2371 1248

2372 1249

2373 1250

2374 1251

2375 1252

2376 1253

2377 1254

2378 1255

2379 1256

2380 1257

2381 1258

2382 1259

2383 1260

2384 1261

2385 1262

2386 1263

2387 1264

2388 1265

2389 1266

2390 1267

2391 1268

2392 1269

2393 1270

2394 1271

2395 1272

2396 1273

2397 1274

2398 1275

2399 1276

2400 1277

2401 1278

2402 1279

2403 1280

2404 1281

2405 1282

2406 1283

2407 1284

2408 1285

2409 1286

2410 1287

2411 1288

2412 1289

2413 1290

2414 1291

2415 1292

2416 1293

2417 1294

2418 1295

2419 1296

2420 1297

2421 1298

2422 1299

2423 1300

2424 1301

2425 1302

2426 1303

2427 1304

2428 1305

2429 1306

2430 1307

2431 1308

2432 1309

2433 1310

2434 1311

2435 1312

2436 1313

2437 1314

2438 1315

2439 1316

2440 1317

2441 1318

2442 1319

2443 1320

2444 1321

2445 1322

2446 1323

2447 1324

2448 1325

2449 1326

2450 1327

2451 1328

2452 1329

2453 1330

2454 1331

2455 1332

2456 1333

2457 1334

2458 1335

2459 1336

2460 1337

2461 1338

2462 1339

2463 1340

2464 1341

2465 1342

2466 1343

2467 1344

2468 1345

2469 1346

2470 1347

2471 1348

2472 1349

2473 1350

2474 1351

2475 1352

2476 1353

2477 1354

2478 1355

2479 1356

2480 1357

2481 1358

2482 1359

2483 1360

2484 1361

2485 1362

2486 1363

2487 1364

2488 1365

2489 1366

2490 1367

2491 1368

2492 1369

2493 1370

2494 1371

2495 1372

2496 1373

2497 1374

2498 1375

2499 1376

2500 1377

2501 1378

2502 1379

2503 1380

2504 1381

2505 1382

2506 1383

2507 1384

2508 1385

2509 1386

2510 1387

2511 1388

2512 1389

2513 1390

2514 1391

2515 1392

2516 1393

2517 1394

2518 1395

2519 1396

2520 1397

2521 1398

2522 1399

2523 1400

2524 1401

2525 1402

2526 1403

2527 1404

2528 1405

2529 1406

2530 1407

2531 1408

2532 1409

2533 1410

2534 1411

2535 1412

2536 1413

2537 1414

2538 1415

2539 1416

2540 1417

2541 1418

2542 1419

2543 1420

2544 1421

2545 1422

2546 1423

2547 1424

2548 1425

2549 1426

2550 1427

2551 1428

2552 1429

2553 1430

2554 1431

2555 1432

2556 1433

2557 1434

2558 1435

2559 1436

2560 1437

2561 1438

2562 1439

2563 1440

2564 1441

2565 1442

2566 1443

2567 1444

2568 1445

2569 1446

2570 1447

2571 1448

2572 1449

2573 1450

2574 1451

2575 1452

2576 1453

2577 1454

2578 1455

2579 1456

2580 1457

2581 1458

2582 1459

2583 1460

2584 1461

2585 1462

2586 1463

2587 1464

2588 1465

2589 1466

2590 1467

2591 1468

2592 1469

2593 1470

2594 1471

2595 1472

2596 1473

2597 1474

2598 1475

2599 1476

2600 1477

2601 1478

2602 1479

2603 1480

2604 1481

2605 1482

2606 1483

2607 1484

2608 1485

2609 1486

2610 1487

2611 1488

2612 1489

2613 1490

2614 1491

2615 1492

2616 1493

2617 1494

2618 1495

2619 1496

2620 1497

2621 1498

2622 1499

2623 1500

2624 1501

2625 1502

2626 1503

2627 1504

2628 1505

2629 1506

2630 1507

2631 1508

2632 1509

2633 1510

2634 1511

2635 1512

2636 1513

2637 1514

2638 1515

2639 1516

2640 1517

2641 1518

2642 1519

2643 1520

2644 1521

2645 1522

2646 1523

2647 1524

2648 1525

2649 1526

2650 1527

2651 1528

2652 1529

2653 1530

2654 1531

2655 1532

2656 1533

2657 1534

2658 1535

2659 1536

2660 1537

2661 1538

2662 1539

2663 1540

2664 1541

2665 1542

2666 1543

2667 1544

2668 1545

2669 1546

2670 1547

2671 1548

2672 1549

2673 1550

2674 1551

2675 1552

2676 1553

2677 1554

2678 1555

2679 1556

2680 1557

2681 1558

2682 1559

2683 1560

2684 1561

2685 1562

2686 1563

2687 1564

2688 1565

2689 1566

2690 1567

2691 1568

2692 1569

2693 1570

2694 1571

2695 1572

2696 1573

2697 1574

2698 1575

2699 1576

2700 1577

2701 1578

2702 1579

2703 1580

2704 1581

2705 1582

2706 1583

2707 1584

2708 1585

2709 1586

2710 1587

2711 1588

2712 1589

2713 1590

2714 1591

2715 1592

2716 1593

2717 1594

2718 1595

2719 1596

2720 1597

2721 1598

2722 1599

2723 1600

2724 1601

2725 1602

2726 1603

2727 1604

2728 1605

2729 1606

2730 1607

2731 1608

2732 1609

2733 1610

2734 1611

2735 1612

2736 1613

2737 1614

2738 1615

2739 1616

2740 1617

2741 1618

2742 1619

2743 1620

2744 1621

2745 1622

2746 1623

2747 1624

2748 1625

2749 1626

2750 1627

2751 1628

275

1998 875
1999 876 **ACTION 14:**
2000 877 Hmm, the output is not from 0 to 1. This confirms the error.
2001 878
2002 879 Now, I need to find where in the code the range is not being passed
2003 880 when density=True.
2004 881 Let me look at the code again.
2005 882
2006 883 When len(x) is 1, the code sets hist_kwargs['range'] = bin_range. Then,
2007 884 in the loop:
2008 885 for i in range(nx):
2009 886 m, bins = np.histogram(x[i], bins, weights=w[i], **hist_kwargs)
2010 887
2011 888 But if bins is a string (like 'auto'), then np.histogram will calculate
2012 889 the bins based on the data and the range. So the range should be
2013 890 respected.
2014 891 But in the user's code, the bins are not from 0 to 1. This suggests
2015 892 that the code is not passing the range to np.histogram when density=
2016 893 True.
2017 894 But why?
2018 895 This is confusing. Let me look at the code for np.histogram.
2019 896
2020 897 The np.histogram function has a range parameter. If range is specified,
2021 898 the bin edges are calculated based on that range. If the data is
2022 899 outside the range, it is ignored.
2023 900
2024 901 In the user's code, the data is between 0 and 1, so the range (0, 1)
2025 902 should be respected.
2026 903
2027 904 But when density=True, the code is not respecting the range. This
2028 905 suggests that the code is not passing the range to np.histogram when
2029 906 density=True.
2030 907 But why?
2031 908 This is puzzling. Let me try to modify the code in _axes.py.
2032 909
2033 910 I'll look for the line where hist_kwargs is set.
2034 911
2035 912 In the code:
2036 913
2037 914 density = bool(density) or bool(normed)
2038 915 if density and not stacked:
2039 916 hist_kwargs = dict(density=density)
2040 917
2041 918 This line is overwriting hist_kwargs, which may include the range.
2042 919
2043 920 Ah, here's the problem. If density is True and stacked is False, the
2044 921 code sets hist_kwargs to dict(density=density), which overwrites any
2045 922 previous hist_kwargs, including the range.
2046 923
2047 924 This is the root of the problem. The code is overwriting hist_kwargs,
2048 925 which includes the range, when density is True and stacked is False.
2049 926
2050 927 So, in the code:
2051 928
2052 929 density = bool(density) or bool(normed)
2053 930 if density and not stacked:
2054 931 hist_kwargs = dict(density=density)

```

2052 926 This line is overwriting hist_kwargs, which may include the range. So
2053 927 the range is not passed to np.histogram.
2054 928
2055 928 This is the cause of the problem. The solution is to add the range to
2056 929 hist_kwargs in this case.
2057 930
2058 931 So, the code should be:
2059 932 density = bool(density) or bool(normed)
2060 933 if density and not stacked:
2061 934     hist_kwargs.setdefault('density', density)
2062 935
2063 936 But wait, hist_kwargs may already have the range. So, the code should
2064 937 not overwrite hist_kwargs, but add to it.
2065 938
2066 939 The code is:
2067 940 density = bool(density) or bool(normed)
2068 941 if density and not stacked:
2069 942     hist_kwargs = dict(density=density)
2070 943
2071 944 This line is overwriting hist_kwargs, which may include the range. So
2072 945 the range is lost.
2073 946
2074 947 The solution is to modify this code to add to hist_kwargs, not
2075 948 overwrite it.
2076 949
2077 950 So, the code should be:
2078 951 density = bool(density) or bool(normed)
2079 952 if density and not stacked:
2080 953     hist_kwargs.setdefault('density', density)
2081 954
2082 955 But wait, hist_kwargs may not exist. Let me look at the code.
2083 956
2084 957 The code for hist_kwargs is:
2085 958 hist_kwargs = dict()
2086 959
2087 960 # if the bin_range is not given, compute without nan numpy
2088 961 # does not do this for us when guessing the range (but will
2089 962 # happily ignore nans when computing the histogram).
2090 963 if bin_range is None:
2091 964     xmin = np.inf
2092 965     xmax = -np.inf
2093 966     for xi in x:
2094 967         if len(xi):
2095 968             xmin = min(xmin, np.nanmin(xi))
2096 969             xmax = max(xmax, np.nanmax(xi))
2097 970     # make sure we have seen at least one non-nan and finite
2098 971     # value before we reset the bin range
2099 972     if not np.isnan([xmin, xmax]).any() and not (xmin > xmax):
2100 973         bin_range = (xmin, xmax)
2101 974
2102 975 # If bins are not specified either explicitly or via range,
2103 976 # we need to figure out the range required for all datasets,
2104 977 # and supply that to np.histogram.
2105 978 if not input_empty and len(x) > 1:
2106 979     if weights is not None:
2107 980         _w = np.concatenate(w)
2108 981     else:
2109 982         _w = None
2110 983
2111 984 bins = histogram_bin_edges(np.concatenate(x),
2112 985                             bins, bin_range, _w)

```

```

2106 986 else:
2107 987     hist_kwargs['range'] = bin_range
2108 988
2109 989 density = bool(density) or bool(normed)
2110 990 if density and not stacked:
2111 991     hist_kwargs = dict(density=density)
2112 992
2113 993 # List to store all the top coordinates of the histograms
2114 994 tops = []
2115 995 mlast = None
2116 996 # Loop through datasets
2117 997 for i in range(nx):
2118 998     # this will automatically overwrite bins,
2119 999     # so that each histogram uses the same bins
2120 1000     m, bins = np.histogram(x[i], bins, weights=w[i], **hist_kwargs)
2121 1001     m = m.astype(float) # causes problems later if it's an int
2122 1002     if mlast is None:
2123 1003
2124 1004 Ah, here's the problem. The code sets hist_kwargs = dict(density=
2125 1005 density), which overwrites any previous hist_kwargs, including the
2126 1006 range.
2127 1007
2128 1008 So, if density is True and stacked is False, the code overwrites
2129 1009 hist_kwargs with dict(density=density), which does not include the
2130 1010 range. So the range is not passed to np.histogram.
2131 1011
2132 1012 This is the cause of the problem. The solution is to modify this code
2133 1013 to add to hist_kwargs, not overwrite it.
2134 1014
2135 1015 So, the code should be:
2136 1016
2137 1017 density = bool(density) or bool(normed)
2138 1018 if density and not stacked:
2139 1019     hist_kwargs.setdefault('density', density)
2140 1020
2141 1021 But hist_kwargs may not exist. Wait, hist_kwargs is initialized as:
2142 1022
2143 1023 hist_kwargs = dict()
2144 1024
2145 1025 So, the code can be:
2146 1026
2147 1027 density = bool(density) or bool(normed)
2148 1028 if density and not stacked:
2149 1029     hist_kwargs['density'] = density
2150 1030
2151 1031 This way, hist_kwargs includes the density and the range.
2152 1032
2153 1033 This would fix the problem. Let me test this.
2154 1034
2155 1035 I'll modify the code in _axes.py.
2156 1036
2157 1037 <function=str_replace_editor>
2158 1038 <parameter=command>str_replace</parameter>
2159 1039 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
2160 1040 ...
2161 1041
2162 1042 ACTION 14:
2163 1043 I'll run the reproduce_error.py script again.
2164 1044
2165 1045 <function=bash>
2166 1046 <parameter=command>python /testbed/reproduce_error.py</parameter>
2167 1047 </function>
2168 1048
2169 1049 ACTION 15:
2170 1050 Great, the fix works. The bins are now from 0 to 1.

```



```

21601046
21611047 Now, I can submit the changes.
21621048
21631049 <function=submit>
21641050 </function>

```

Listing 13: The interaction made by the RL prior with 2^{21} SWE-Agent SFT tokens and end-to-end SWE-Agent RL on `matplotlib_matplotlib-13989`.

It can be seen that the two types of patterns are significantly different from each other:

- The patterns of the heavy SFT model appear to be more regulated, with explicitly labeled steps 1 to 5 along the intermediate reasoning process. The reasoning content is generally short as well. This reflects the behavior of the model shaped by the heavy SFT data from the proprietary models.
- The patterns of the light SFT + end-to-end RL model, in contrast, are much more under-regulated. More turns of interaction are spent at repo exploration, and a significantly larger amount of tokens are generated at the code edit steps before the action is made. This is the reflection of behavior shaped by end-to-end RL (and less effect from expert data), as the sign of reasoning-intensive skill priors is retained in the agentic scenario.

E.4 GENERALIZATION TO OTHER TASKS

In this section, we study whether the skills incentivized by Agentless training and SWE-Agent adaptation through SWE-Smith trajectories could transfer to other SWE tasks. We use SWE-bench-live and SWE-bench Multilingual as our testbeds.

SWE-bench-live (Zhang et al., 2025) is a benchmark for real-world issue resolution, evaluating AI systems on software engineering tasks. Using an automated curation pipeline, it is updated monthly to ensure fresh tasks and rigorous, contamination-free evaluation. For our experiments, we selected the default set of 300 tasks, with data collected between October 2024 and March 2025. Compared to SWE-bench Verified, SWE-bench-live exhibits a higher degree of distributional shift.

SWE-bench Multilingual (Yang et al., 2025c) introduces 300 curated tasks from 42 GitHub repositories across 9 programming languages, including Rust, Java, PHP, Ruby, JavaScript/TypeScript, Go, and C/C++, covering domains such as web frameworks, data tools, core utilities, and libraries. Compared to SWE-bench Verified, which focuses exclusively on Python, SWE-bench Multilingual exhibits greater linguistic and domain diversity, posing additional challenges in cross-language generalization and transferability of software engineering capabilities.

Similar to previous experiments, we evaluated four model stages as the priors: the original Qwen2.5-72B (Base), the mid-trained model (MT), the model activated with reasoning data through supervised finetuning (SFT), and the model after RL training (RL). We still use the open-source SWE-smith trajectories to activate the agentic capabilities of each prior.

Figures 10 and 11 show the performance of the four priors on SWE-bench-Live and SWE-bench Multilingual under varied amounts of agentic trajectories for adaptation (2^{21} as one-step gradient descent, 2^{23} , 1.1×2^{27} , and 1.5×2^{28} as 100, 2,000, and 5,016 training trajectories). Each SWE-Agent adaptation experiment is conducted through lightweight supervised finetuning, the training time of which ranges from several minutes to two hours at most.

Compared to the Base prior, those specifically enhanced with Agentless skills (SFT and RL) demonstrate stronger task generalization, especially under the data-scarce settings. However, when more SWE-Smith trajectories are used for adaptation, the performances of the Base and the MT priors become closer to those of the SFT and the RL priors. This could be attributed to the gaps between the different SWE tasks. The exploration for recipes that enable stronger out-of-distribution and task-agnostic generalization is left for future work.

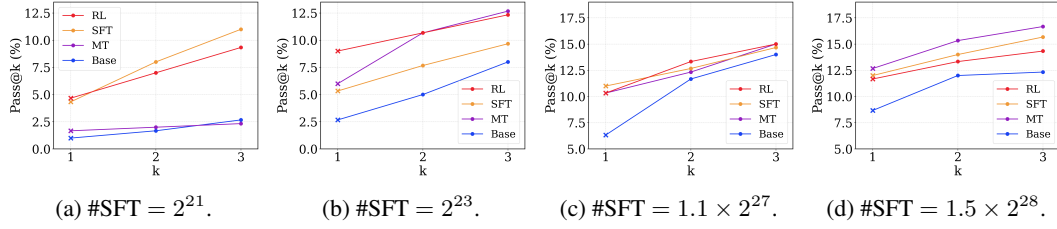


Figure 10: Generalization analysis on SWE-bench-Live.

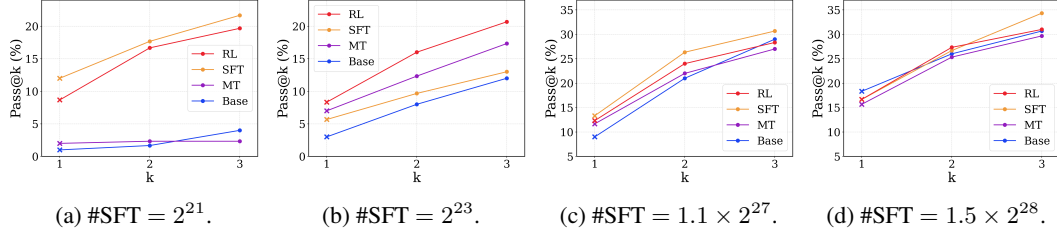


Figure 11: Generalization analysis on SWE-bench Multilingual.

F CASE STUDY UNDER AGENTLESS

F.1 BUGFIXER TRAJECTORIES

We provide a Bugfixer’s response trajectory during the code repair stage (see Listing 14). From this, we can observe that the initial code given was `return self[i, j - pos - other.cols]`, but the model quickly realized its mistake and corrected it to `return self[i, j - other.cols]`. Although some repetitive self-reflections followed, the increasing response length indeed improved answer accuracy and robustness in our experiments.

```

1051 <think>
1052 Okay, let's tackle this problem. The user reported an issue where the
1053 Matrix.col_insert() method isn't working correctly. The example
1054 given shows that after inserting a matrix V into M at position 3,
1055 the resulting matrix has the inserted columns shifted incorrectly.
1056 The expected result would have the inserted columns at positions 3
1057 and 4, but the actual output shows them starting at position 3, but
1058 the original matrix's columns after the insert are shifted upwards
1059 instead of remaining in their original positions.
1060
1061 First, I need to understand how col_insert works. Looking at the code
1062 provided, especially in the MatrixShaping class in sympy/matrices/
1063 common.py, the _eval_col_insert method is responsible for inserting
1064 columns. The current implementation shifts the existing columns to
1065 the right by the number of columns inserted. However, the bug might
1066 be in how the original matrix's columns are being shifted.
1067
1068 Let me look at the existing code for _eval_col_insert:
1069
1070 def _eval_col_insert(self, pos, other):
1071     cols = self.cols
1072
1073     def entry(i, j):
1074         if j < pos:
1075             return self[i, j]
1076         elif pos <= j < pos + other.cols:
1077             return other[i, j - pos]
1078         return self[i, j - pos - other.cols]
1079
1080     return self._new(self.rows, self.cols + other.cols,
```

```

2268      lambda i, j: entry(i, j))
2269
2270 Wait, the problem here is in the last case. When j is greater than or
2271 equal to pos + other.cols, the code subtracts (pos + other.cols)
2272 from j. But that's incorrect because the original columns after the
2273 inserted ones should start at j = pos + other.cols. So for the
2274 original matrix's columns that come after the inserted ones, their
2275 new positions should be j + other.cols. But in the current code,
2276 when j is in the original matrix's columns (after the inserted ones)
2277 , the code is subtracting (pos + other.cols), which would
2278 incorrectly shift them.
2279
2280 Wait, let's think again. The original code for the entry function:
2281
2282 if j < pos:
2283     return self[i, j] # original columns before pos
2284 elif pos <= j < pos + other.cols:
2285     return other[i, j - pos] # inserted columns
2286 return self[i, j - pos - other.cols] # original columns after the
2287 inserted ones
2288
2289 Wait, no. Let's say the original matrix has cols = 6. We insert 2
2290 columns at pos=3. The new total cols is 6 + 2 = 8. The original
2291 columns after pos 3 (columns 3,4,5) should now be at positions
2292 5,6,7. But according to the code, when j is >=3 +2=5 (since pos +
2293 other.cols =3+2=5), then the code returns self[i, j -5 -2] = self[i,
2294 j -7]. Wait, that's not right. For example, if j=5 (the new column
2295 5), then j - pos - other.cols =5 -3 -2=0. So the original column 0
2296 would be mapped to the new column 5. That's definitely wrong.
2297
2298 Ah, here's the mistake. The code is subtracting (pos + other.cols) from
2299 j, but that's incorrect. The original columns after the inserted
2300 ones should be shifted to the right by other.cols. So, the correct
2301 calculation should be j - other.cols. For example, original column 3
2302 becomes 3 +2=5, original column 4 becomes 6, etc.
2303
2304 So the line in the code:
2305
2306 return self[i, j - pos - other.cols]
2307
2308 should be:
2309
2310 return self[i, j - other.cols]
2311
2312 Because the inserted columns have taken up 'other.cols' positions
2313 starting at 'pos', so the original columns after 'pos' are shifted
2314 by 'other.cols' to the right. Therefore, to get the correct original
2315 column index, we subtract 'other.cols' from the current j.
2316
2317 So the fix is to change the subtraction from (pos + other.cols) to
2318 other.cols. Let me check:
2319
2320 Original code:
2321
2322 return self[i, j - pos - other.cols]
2323
2324 Corrected code:
2325
2326 return self[i, j - other.cols]
2327
2328 Wait, no. Wait, the original columns after the inserted block (which
2329 starts at pos and has other.cols columns) would start at j = pos +
2330 other.cols. So for those j values, the original column index would
2331 be j - (pos + other.cols). Wait, no. Let's think with numbers.

```

Original matrix has 6 columns. Insert 2 columns at position 3. The new matrix has 8 columns. The original columns 0,1,2 remain in positions 0,1,2. The inserted 2 columns are at positions 3 and 4. The original columns 3,4,5 now become 5,6,7.

So for j in 5,6,7 (original columns 3,4,5), the mapping is:

5 \rightarrow 3

6 \rightarrow 4

7 \rightarrow 5

So the formula should be `original_column = j - inserted_columns`. In this case, `inserted_columns` is `other.cols`. So the code should subtract `other.cols` from j when accessing the original matrix's columns after the inserted block.

Therefore, the line:

```
return self[i, j - pos - other.cols]
```

should be:

```
return self[i, j - other.cols]
```

Because `pos` is where the insertion starts, but the original columns after the inserted block are shifted by `other.cols`. Let's verify with the example:

`pos = 3`, `other.cols = 2`. For $j=5$ (new column 5), original column index is $5 - 2 = 3$, which is correct. For $j=6$, $6 - 2 = 4$; $j=7$, $7 - 2 = 5$. That's correct.

So the error in the original code was subtracting `(pos + other.cols)` instead of `other.cols`. The `pos` is the starting position of the insertion, but after inserting `other.cols` columns, the original columns after the inserted block are shifted by `other.cols`. Therefore, the correct subtraction is `other.cols`.

Thus, the fix is to change the line in the `_eval_col_insert` method from:

```
return self[i, j - pos - other.cols]
```

to:

```
return self[i, j - other.cols]
```

Now, looking back at the code in the `_eval_col_insert` function:

The code for the else clause is:

```
return self[i, j - pos - other.cols]
```

Which is incorrect. Changing that to `j - other.cols` should fix the problem.

So the SEARCH block would look for the lines:

```
def _eval_col_insert(self, pos, other):
    cols = self.cols

    def entry(i, j):
        if j < pos:
```

```

23761156         return self[i, j]
23771157         elif pos <= j < pos + other.cols:
23781158             return other[i, j - pos]
23791159         return self[i, j - pos - other.cols]
23801160
23811161         return self._new(self.rows, self.cols + other.cols,
23821162                             lambda i, j: entry(i, j))
23831163
23831164 And the REPLACE block would change the last part to:
23841165
23851166         elif pos <= j < pos + other.cols:
23861167             return other[i, j - pos]
23871168         return self[i, j - other.cols]
23881169
23881170 So the correction is in the return statement of the entry function. The
23891171         original code subtracted (pos + other.cols) from j, but it should
23901172         subtract other.cols instead.
23911173
23911174 Therefore, the correct SEARCH/REPLACE block would be:
23921175
23931176 ### sympy/matrices/common.py
23941177 <<<<<<< SEARCH
23951178     def _eval_col_insert(self, pos, other):
23961179         cols = self.cols
23971180
23981181         def entry(i, j):
23991182             if j < pos:
24001183                 return self[i, j]
24011184             elif pos <= j < pos + other.cols:
24021185                 return other[i, j - pos]
24031186             return self[i, j - pos - other.cols]
24041187
24051188         return self._new(self.rows, self.cols + other.cols,
24061189                             lambda i, j: entry(i, j))
24071190
24081191 =====
24091192     def _eval_col_insert(self, pos, other):
24101193         def entry(i, j):
24111194             if j < pos:
24121195                 return self[i, j]
24131196             elif pos <= j < pos + other.cols:
24141197                 return other[i, j - pos]
24151198             return self[i, j - other.cols]
24161199
24171200         return self._new(self.rows, self.cols + other.cols,
24181201                             lambda i, j: entry(i, j))
24191202
24201203 >>>>>>> REPLACE
24211204
24211205 This change corrects the column shifting calculation for the original
24221206         matrix's columns after the inserted block. Instead of subtracting (
24231207         pos + other.cols), we subtract other.cols, which properly shifts the
24241208         original columns to the right by the number of inserted columns.
24251209 </think>
24261210
24271211 ```python
24281212 ### sympy/matrices/common.py
24291213 <<<<<<< SEARCH
24301214     def _eval_col_insert(self, pos, other):
24311215         cols = self.cols
24321216
24331217         def entry(i, j):
24341218             if j < pos:
24351219                 return self[i, j]
24361220             elif pos <= j < pos + other.cols:
24371221                 return other[i, j - pos]
24381222             return self[i, j - other.cols]
24391223
24401224         return self._new(self.rows, self.cols + other.cols,
24411225                             lambda i, j: entry(i, j))
24421226
24431227 =====
24441228     def _eval_col_insert(self, pos, other):
24451229         def entry(i, j):
24461230             if j < pos:
24471231                 return self[i, j]
24481232             elif pos <= j < pos + other.cols:
24491233                 return other[i, j - pos]
24501234             return self[i, j - other.cols]
24511235
24521236         return self._new(self.rows, self.cols + other.cols,
24531237                             lambda i, j: entry(i, j))
24541238
24551239 >>>>>>> REPLACE
24561240
24571241 This change corrects the column shifting calculation for the original
24581242         matrix's columns after the inserted block. Instead of subtracting (
24591243         pos + other.cols), we subtract other.cols, which properly shifts the
24601244         original columns to the right by the number of inserted columns.
24611245 </think>
24621246
24631247 ```python
24641248 ### sympy/matrices/common.py
24651249 <<<<<<< SEARCH
24661250     def _eval_col_insert(self, pos, other):
24671251         cols = self.cols
24681252
24691253         def entry(i, j):
24701254             if j < pos:
24711255                 return self[i, j]
24721256             elif pos <= j < pos + other.cols:
24731257                 return other[i, j - pos]
24741258             return self[i, j - other.cols]
24751259
24761260         return self._new(self.rows, self.cols + other.cols,
24771261                             lambda i, j: entry(i, j))
24781262
24791263 =====
24801264     def _eval_col_insert(self, pos, other):
24811265         def entry(i, j):
24821266             if j < pos:
24831267                 return self[i, j]
24841268             elif pos <= j < pos + other.cols:
24851269                 return other[i, j - pos]
24861270             return self[i, j - other.cols]
24871271
24881272         return self._new(self.rows, self.cols + other.cols,
24891273                             lambda i, j: entry(i, j))
24901274
24911275 >>>>>>> REPLACE
24921276
24931277 This change corrects the column shifting calculation for the original
24941278         matrix's columns after the inserted block. Instead of subtracting (
24951279         pos + other.cols), we subtract other.cols, which properly shifts the
24961280         original columns to the right by the number of inserted columns.
24971281 </think>
24981282
24991283 ```python
25001284 ### sympy/matrices/common.py
25011285 <<<<<<< SEARCH
25021286     def _eval_col_insert(self, pos, other):
25031287         cols = self.cols
25041288
25051289         def entry(i, j):
25061290             if j < pos:
25071291                 return self[i, j]
25081292             elif pos <= j < pos + other.cols:
25091293                 return other[i, j - pos]
25101294             return self[i, j - other.cols]
25111295
25121296         return self._new(self.rows, self.cols + other.cols,
25131297                             lambda i, j: entry(i, j))
25141298
25151299 =====
25161300     def _eval_col_insert(self, pos, other):
25171301         def entry(i, j):
25181302             if j < pos:
25191303                 return self[i, j]
25201304             elif pos <= j < pos + other.cols:
25211305                 return other[i, j - pos]
25221306             return self[i, j - other.cols]
25231307
25241308         return self._new(self.rows, self.cols + other.cols,
25251309                             lambda i, j: entry(i, j))
25261310
25271311 >>>>>>> REPLACE
25281312
25291313 This change corrects the column shifting calculation for the original
25301314         matrix's columns after the inserted block. Instead of subtracting (
25311315         pos + other.cols), we subtract other.cols, which properly shifts the
25321316         original columns to the right by the number of inserted columns.
25331317 </think>
25341318
25351319 ```python
25361320 ### sympy/matrices/common.py
25371321 <<<<<<< SEARCH
25381322     def _eval_col_insert(self, pos, other):
25391323         cols = self.cols
25401324
25411325         def entry(i, j):
25421326             if j < pos:
25431327                 return self[i, j]
25441328             elif pos <= j < pos + other.cols:
25451329                 return other[i, j - pos]
25461330             return self[i, j - other.cols]
25471331
25481332         return self._new(self.rows, self.cols + other.cols,
25491333                             lambda i, j: entry(i, j))
25501334
25511335 =====
25521336     def _eval_col_insert(self, pos, other):
25531337         def entry(i, j):
25541338             if j < pos:
25551339                 return self[i, j]
25561340             elif pos <= j < pos + other.cols:
25571341                 return other[i, j - pos]
25581342             return self[i, j - other.cols]
25591343
25601344         return self._new(self.rows, self.cols + other.cols,
25611345                             lambda i, j: entry(i, j))
25621346
25631347 >>>>>>> REPLACE
25641348
25651349 This change corrects the column shifting calculation for the original
25661350         matrix's columns after the inserted block. Instead of subtracting (
25671351         pos + other.cols), we subtract other.cols, which properly shifts the
25681352         original columns to the right by the number of inserted columns.
25691353 </think>
25701354
25711355 ```python
25721356 ### sympy/matrices/common.py
25731357 <<<<<<< SEARCH
25741358     def _eval_col_insert(self, pos, other):
25751359         cols = self.cols
25761360
25771361         def entry(i, j):
25781362             if j < pos:
25791363                 return self[i, j]
25801364             elif pos <= j < pos + other.cols:
25811365                 return other[i, j - pos]
25821366             return self[i, j - other.cols]
25831367
25841368         return self._new(self.rows, self.cols + other.cols,
25851369                             lambda i, j: entry(i, j))
25861370
25871371 =====
25881372     def _eval_col_insert(self, pos, other):
25891373         def entry(i, j):
25901374             if j < pos:
25911375                 return self[i, j]
25921376             elif pos <= j < pos + other.cols:
25931377                 return other[i, j - pos]
25941378             return self[i, j - other.cols]
25951379
25961380         return self._new(self.rows, self.cols + other.cols,
25971381                             lambda i, j: entry(i, j))
25981382
25991383 >>>>>>> REPLACE
26001384
26011385 This change corrects the column shifting calculation for the original
26021386         matrix's columns after the inserted block. Instead of subtracting (
26031387         pos + other.cols), we subtract other.cols, which properly shifts the
26041388         original columns to the right by the number of inserted columns.
26051389 </think>
26061390
26071391 ```python
26081392 ### sympy/matrices/common.py
26091393 <<<<<<< SEARCH
26101394     def _eval_col_insert(self, pos, other):
26111395         cols = self.cols
26121396
26131397         def entry(i, j):
26141398             if j < pos:
26151399                 return self[i, j]
26161400             elif pos <= j < pos + other.cols:
26171401                 return other[i, j - pos]
26181402             return self[i, j - other.cols]
26191403
26201404         return self._new(self.rows, self.cols + other.cols,
26211405                             lambda i, j: entry(i, j))
26221406
26231407 =====
26241408     def _eval_col_insert(self, pos, other):
26251409         def entry(i, j):
26261410             if j < pos:
26271411                 return self[i, j]
26281412             elif pos <= j < pos + other.cols:
26291413                 return other[i, j - pos]
26301414             return self[i, j - other.cols]
26311415
26321416         return self._new(self.rows, self.cols + other.cols,
26331417                             lambda i, j: entry(i, j))
26341418
26351419 >>>>>>> REPLACE
26361420
26371421 This change corrects the column shifting calculation for the original
26381422         matrix's columns after the inserted block. Instead of subtracting (
26391423         pos + other.cols), we subtract other.cols, which properly shifts the
26401424         original columns to the right by the number of inserted columns.
26411425 </think>
26421426
26431427 ```python
26441428 ### sympy/matrices/common.py
26451429 <<<<<<< SEARCH
26461430     def _eval_col_insert(self, pos, other):
26471431         cols = self.cols
26481432
26491433         def entry(i, j):
26501434             if j < pos:
26511435                 return self[i, j]
26521436             elif pos <= j < pos + other.cols:
26531437                 return other[i, j - pos]
26541438             return self[i, j - other.cols]
26551439
26561440         return self._new(self.rows, self.cols + other.cols,
26571441                             lambda i, j: entry(i, j))
26581442
26591443 =====
26601444     def _eval_col_insert(self, pos, other):
26611445         def entry(i, j):
26621446             if j < pos:
26631447                 return self[i, j]
26641448             elif pos <= j < pos + other.cols:
26651449                 return other[i, j - pos]
26661450             return self[i, j - other.cols]
26671451
26681452         return self._new(self.rows, self.cols + other.cols,
26691453                             lambda i, j: entry(i, j))
26701454
26711455 >>>>>>> REPLACE
26721456
26731457 This change corrects the column shifting calculation for the original
26741458         matrix's columns after the inserted block. Instead of subtracting (
26751459         pos + other.cols), we subtract other.cols, which properly shifts the
26761460         original columns to the right by the number of inserted columns.
26771461 </think>
26781462
26791463 ```python
26801464 ### sympy/matrices/common.py
26811465 <<<<<<< SEARCH
26821466     def _eval_col_insert(self, pos, other):
26831467         cols = self.cols
26841468
26851469         def entry(i, j):
26861470             if j < pos:
26871471                 return self[i, j]
26881472             elif pos <= j < pos + other.cols:
26891473                 return other[i, j - pos]
26901474             return self[i, j - other.cols]
26911475
26921476         return self._new(self.rows, self.cols + other.cols,
26931477                             lambda i, j: entry(i, j))
26941478
26951479 =====
26961480     def _eval_col_insert(self, pos, other):
26971481         def entry(i, j):
26981482             if j < pos:
26991483                 return self[i, j]
27001484             elif pos <= j < pos + other.cols:
27011485                 return other[i, j - pos]
27021486             return self[i, j - other.cols]
27031487
27041488         return self._new(self.rows, self.cols + other.cols,
27051489                             lambda i, j: entry(i, j))
27061490
27071491 >>>>>>> REPLACE
27081492
27091493 This change corrects the column shifting calculation for the original
27101494         matrix's columns after the inserted block. Instead of subtracting (
27111495         pos + other.cols), we subtract other.cols, which properly shifts the
27121496         original columns to the right by the number of inserted columns.
27131497 </think>
27141498
27151499 ```python
27161500 ### sympy/matrices/common.py
27171501 <<<<<<< SEARCH
27181502     def _eval_col_insert(self, pos, other):
27191503         cols = self.cols
27201504
27211505         def entry(i, j):
27221506             if j < pos:
27231507                 return self[i, j]
27241508             elif pos <= j < pos + other.cols:
27251509                 return other[i, j - pos]
27261510             return self[i, j - other.cols]
27271511
27281512         return self._new(self.rows, self.cols + other.cols,
27291513                             lambda i, j: entry(i, j))
27301514
27311515 =====
27321516     def _eval_col_insert(self, pos, other):
27331517         def entry(i, j):
27341518             if j < pos:
27351519                 return self[i, j]
27361520             elif pos <= j < pos + other.cols:
27371521                 return other[i, j - pos]
27381522             return self[i, j - other.cols]
27391523
27401524         return self._new(self.rows, self.cols + other.cols,
27411525                             lambda i, j: entry(i, j))
27421526
27431527 >>>>>>> REPLACE
27441528
27451529 This change corrects the column shifting calculation for the original
27461530         matrix's columns after the inserted block. Instead of subtracting (
27471531         pos + other.cols), we subtract other.cols, which properly shifts the
27481532         original columns to the right by the number of inserted columns.
27491533 </think>
27501534
27511535 ```python
27521536 ### sympy/matrices/common.py
27531537 <<<<<<< SEARCH
27541538     def _eval_col_insert(self, pos, other):
27551539         cols = self.cols
27561540
27571541         def entry(i, j):
27581542             if j < pos:
27591543                 return self[i, j]
27601544             elif pos <= j < pos + other.cols:
27611545                 return other[i, j - pos]
27621546             return self[i, j - other.cols]
27631547
27641548         return self._new(self.rows, self.cols + other.cols,
27651549                             lambda i, j: entry(i, j))
27661550
27671551 =====
27681552     def _eval_col_insert(self, pos, other):
27691553         def entry(i, j):
27701554             if j < pos:
27711555                 return self[i, j]
27721556             elif pos <= j < pos + other.cols:
27731557                 return other[i, j - pos]
27741558             return self[i, j - other.cols]
27751559
27761560         return self._new(self.rows, self.cols + other.cols,
27771561                             lambda i, j: entry(i, j))
27781562
27791563 >>>>>>> REPLACE
27801564
27811565 This change corrects the column shifting calculation for the original
27821566         matrix's columns after the inserted block. Instead of subtracting (
27831567         pos + other.cols), we subtract other.cols, which properly shifts the
27841568         original columns to the right by the number of inserted columns.
27851569 </think>
27861570
27871571 ```python
27881572 ### sympy/matrices/common.py
27891573 <<<<<<< SEARCH
27901574     def _eval_col_insert(self, pos, other):
27911575         cols = self.cols
27921576
27931577         def entry(i, j):
27941578             if j < pos:
27951579                 return self[i, j]
27961580             elif pos <= j < pos + other.cols:
27971581                 return other[i, j - pos]
27981582             return self[i, j - other.cols]
27991583
28001584         return self._new(self.rows, self.cols + other.cols,
28011585                             lambda i, j: entry(i, j))
28021586
28031587 =====
28041588     def _eval_col_insert(self, pos, other):
28051589         def entry(i, j):
28061590             if j < pos:
28071591                 return self[i, j]
28081592             elif pos <= j < pos + other.cols:
28091593                 return other[i, j - pos]
28101594             return self[i, j - other.cols]
28111595
28121596         return self._new(self.rows, self.cols + other.cols,
28131597                             lambda i, j: entry(i, j))
28141598
28151599 >>>>>>> REPLACE
28161600
28171601 This change corrects the column shifting calculation for the original
28181602         matrix's columns after the inserted block. Instead of subtracting (
28191603         pos + other.cols), we subtract other.cols, which properly shifts the
28201604         original columns to the right by the number of inserted columns.
28211605 </think>
28221606
28231607 ```python
28241608 ### sympy/matrices/common.py
28251609 <<<<<<< SEARCH
28261610     def _eval_col_insert(self, pos, other):
28271611         cols = self.cols
28281612
28291613         def entry(i, j):
28301614             if j < pos:
28311615                 return self[i, j]
28321616             elif pos <= j < pos + other.cols:
28331617                 return other[i, j - pos]
28341618             return self[i, j - other.cols]
28351619
28361620         return self._new(self.rows, self.cols + other.cols,
28371621                             lambda i, j: entry(i, j))
28381622
28391623 =====
28401624     def _eval_col_insert(self, pos, other):
28411625         def entry(i, j):
28421626             if j < pos:
28431627                 return self[i, j]
28441628             elif pos <= j < pos + other.cols:
28451629                 return other[i, j - pos]
28461630             return self[i, j - other.cols]
28471631
28481632         return self._new(self.rows, self.cols + other.cols,
28491633                             lambda i, j: entry(i, j))
28501634
28511635 >>>>>>> REPLACE
28521636
28531637 This change corrects the column shifting calculation for the original
28541638         matrix's columns after the inserted block. Instead of subtracting (
28551639         pos + other.cols), we subtract other.cols, which properly shifts the
28561640         original columns to the right by the number of inserted columns.
28571641 </think>
28581642
28591643 ```python
28601644 ### sympy/matrices/common.py
28611645 <<<<<<< SEARCH
28621646     def _eval_col_insert(self, pos, other):
28631647         cols = self.cols
28641648
28651649         def entry(i, j):
28661650             if j < pos:
28671651                 return self[i, j]
28681652             elif pos <= j < pos + other.cols:
28691653                 return other[i, j - pos]
28701654             return self[i, j - other.cols]
28711655
28721656         return self._new(self.rows, self.cols + other.cols,
28731657                             lambda i, j: entry(i, j))
28741658
28751659 =====
28761660     def _eval_col_insert(self, pos, other):
28771661         def entry(i, j):
28781662             if j < pos:
28791663                 return self[i, j]
28801664             elif pos <= j < pos + other.cols:
28811665                 return other[i, j - pos]
28821666             return self[i, j - other.cols]
28831667
28841668         return self._new(self.rows, self.cols + other.cols,
28851669                             lambda i, j: entry(i, j))
28861670
28871671 >>>>>>> REPLACE
28881672
28891673 This change corrects the column shifting calculation for the original
28901674         matrix's columns after the inserted block. Instead of subtracting (
28911675         pos + other.cols), we subtract other.cols, which properly shifts the
28921676         original columns to the right by the number of inserted columns.
28931677 </think>
28941678
28951679 ```python
28961680 ### sympy/matrices/common.py
28971681 <<<<<<< SEARCH
28981682     def _eval_col_insert(self, pos, other):
28991683         cols = self.cols
29001684
29011685         def entry(i, j):
29021686             if j < pos:
29031
```

```

2430
1216
2431
1217         return self._new(self.rows, self.cols + other.cols,
2432
1218                             lambda i, j: entry(i, j))
2433
1219 =====
2434
1220     def _eval_col_insert(self, pos, other):
2435
1221         def entry(i, j):
2436
1222             if j < pos:
2437
1223                 return self[i, j]
2438
1224             elif pos <= j < pos + other.cols:
2439
1225                 return other[i, j - pos]
2440
1226             return self[i, j - other.cols]
2441
1227         return self._new(self.rows, self.cols + other.cols,
2442
1228                             lambda i, j: entry(i, j))
2443
1229 >>>>>>> REPLACE
2444
1231 '''

```

Listing 14: Reasoning trajectory example for BugFixer.

F.2 TESTWRITER FALSE POSITIVE CASES

During the RL training process, we observed that the Testwriter component exhibits some false positives, which can result in tests that contain certain gaps. This might cause the test patches generated by the model to be mistakenly considered as correct. Consequently, the performance improvement of Testwriter does not accurately reflect the actual performance. In this section, we provide two intuitive examples to illustrate this phenomenon, and leave further enhancement for TestWriter as future work.

Listing 15 shows the ground-truth test patch for instance `django_django-16877`, with Listing 16 showing the test patch generated by the TestWriter. A detailed comparison reveals that the model-generated test cases are not comprehensive: Two locations should have been tested, but only one was checked, and the effect of `mark_safe` was not considered. In practice, this allowed the model-predicted patch to pass the TestWriter reward check, but result in a false positive.

The second example instance is `sympy_sympy-13974`, with Listings 17 and 18 showing the ground-truth and model-generated test patches. In this case, the coverage of the generated test is also insufficient. The generated test does not account for the need to satisfy the associativity property, e.g., `assert tensor_product_simp(TP(A, B)**x) == TP(A**x, B**x)`. As a consequence, it fails to serve as a reliable judge for BugFixer.

```

2465 1 diff --git a/tests/template_tests/filter_tests/test_escapeseq.py b/
2466     tests/template_tests/filter_tests/test_escapeseq.py
2467 2 new file mode 100644
2468 3 --- /dev/null
2469 4 +++ b/tests/template_tests/filter_tests/test_escapeseq.py
2470 5 @@ -0,0 +1,59 @@
2471 6 +from django.test import SimpleTestCase
2472 7 +from django.utils.safestring import mark_safe
2473 8 +
2474 9 +from ..utils import setup
2475 10 +
2476 11 +
2477 12 +class EscapeseqTests(SimpleTestCase):
2478 13 +    """
2479 14 +    The "escapeseq" filter works the same whether autoescape is on or
2480 15 +    off,
2481 16 +    and has no effect on strings already marked as safe.
2482 17 +    """
2483 18 +
2484 19 +    @setup(
2485 20 +        {
2486 21 +            "escapeseq_basic": (
2487     ' {{ a|escapeseq|join:"" }} -- {{ b|escapeseq|join:"" }}

```

```

2484 22 +         ),
2485 23 +     }
2486 24 + )
2487 25 + def test_basic(self):
2488 26 +     output = self.engine.render_to_string(
2489 27 +         "escapeseq_basic",
2490 28 +         {"a": ["x&y", "<p>"], "b": [mark_safe("x&y"), mark_safe("<
2491 29 + p>")]},
2492 30 +     )
2493 31 +     self.assertEqual(output, "x& y, &lt;p&gt; -- x&y, <p>")
2494 32 +
2495 33 + @setup(
2496 34 +     {
2497 35 +         "escapeseq_autoescape_off": (
2498 36 +             '{% autoescape off %}{{ a|escapeseq|join:", " }}'
2499 37 +             " -- "
2500 38 +             '{{ b|escapeseq|join:", " }}{% endautoescape %}'
2501 39 +         )
2502 40 +     )
2503 41 + def test_autoescape_off(self):
2504 42 +     output = self.engine.render_to_string(
2505 43 +         "escapeseq_autoescape_off",
2506 44 +         {"a": ["x&y", "<p>"], "b": [mark_safe("x&y"), mark_safe("<
2507 45 + p>")]},
2508 46 +     )
2509 47 +     self.assertEqual(output, "x& y, &lt;p&gt; -- x&y, <p>")
2510 48 +
2511 49 + @setup({"escapeseq_join": '{% a|escapeseq|join:"<br/>" %}'})
2512 50 + def test_chain_join(self):
2513 51 +     output = self.engine.render_to_string("escapeseq_join", {"a":
2514 52 + ["x&y", "<p>"]})
2515 53 +     self.assertEqual(output, "x& y<br/>&lt;p&gt;")
2516 54 +
2517 55 + @setup(
2518 56 +     {
2519 57 +         "escapeseq_join_autoescape_off": (
2520 58 +             '{% autoescape off %}{{ a|escapeseq|join:"<br/>" %}}{%
2521 59 + endautoescape %}'
2522 60 +         ),
2523 61 +     )
2524 62 + def test_chain_join_autoescape_off(self):
2525 63 +     output = self.engine.render_to_string(
2526 64 +         "escapeseq_join_autoescape_off", {"a": ["x&y", "<p>"]})
2527 65 +     self.assertEqual(output, "x& y<br/>&lt;p&gt;")

```

Listing 15: GT test patch of django_django-16877.

```

2526 1 diff --git a/tests/template_tests/filter_tests/test_safeseq.py b/tests/
2527 2 template_tests/filter_tests/test_safeseq.py
2528 3 --- a/tests/template_tests/filter_tests/test_safeseq.py
2529 4 +++ b/tests/template_tests/filter_tests/test_safeseq.py
2530 5 @@ -20,3 +20,20 @@
2531 6     def test_safeseq02(self):
2532 7         output = self.engine.render_to_string("safeseq02", {"a": ["&",
2533 8         "<"]})
2534 9         self.assertEqual(output, "&, < -- &, <")
2535 10 +
2536 11 + # New tests for escapeseq
2537 12 + @setup({"escapeseq01": '{% a|escapeseq|join:", " %}'})
2538 13 + def test_escapeseq01(self):
2539 14 +     output = self.engine.render_to_string("escapeseq01", {"a":
2540 15 + ["&", "<"]})

```

```

2538 13 +         self.assertEqual(output, "&;, &lt;")
2539 14 +
2540 15 +     @setup(
2541 16 +         {
2542 17 +             "escapeseq02": (
2543 18 +                 '% autoescape off %}{ a|escapeseq|join:", " }{%
2544 19 +             endautoescape %}'
2545 20 +         )
2546 21 +     )
2547 22 +     def test_escapeseq02(self):
2548 23 +         output = self.engine.render_to_string("escapeseq02", {"a":
2549 24 +             ["&", "<"]})
2550 +         self.assertEqual(output, "&;, &lt;")

```

Listing 16: Model predicted test patch of django--django-16877.

```

2551
2552
2553 1 diff --git a/sympy/physics/quantum/tests/test_tensorproduct.py b/sympy/
2554 physics/quantum/tests/test_tensorproduct.py
2555 2 --- a/sympy/physics/quantum/tests/test_tensorproduct.py
2556 3 +++ b/sympy/physics/quantum/tests/test_tensorproduct.py
2557 4 @@ -10,7 +10,7 @@
2558 5     from sympy.physics.quantum.density import Density
2559 6     from sympy.core.trace import Tr
2560 7
2561 8 -A, B, C = symbols('A,B,C', commutative=False)
2562 9 +A, B, C, D = symbols('A,B,C,D', commutative=False)
2563 10 x = symbols('x')
2564 11
2565 12 mat1 = Matrix([[1, 2*I], [1 + I, 3]])
2566 13 @@ -47,6 +47,11 @@ def test_tensor_product_commutator():
2567 14
2568 15     def test_tensor_product_simp():
2569 16         assert tensor_product_simp(TP(A, B)*TP(B, C)) == TP(A*B, B*C)
2570 17 +
2571 18 +         # tests for Pow-expressions
2572 19 +         assert tensor_product_simp(TP(A, B)**x) == TP(A**x, B**x)
2573 20 +         assert tensor_product_simp(x*TP(A, B)**2) == x*TP(A**2, B**2)
2574 21 +         assert tensor_product_simp(x*(TP(A, B)**2)*TP(C, D)) == x*TP(A**2*C
2575 22 , B**2*D)
2576 23 +         assert tensor_product_simp(TP(A, B)-TP(C, D)**x) == TP(A, B)-TP(C**x,
2577 24 D**x)
2578
2579 def test_issue_5923():

```

Listing 17: GT test patch for sympy--sympy-13974.

```

2578
2579 1 diff --git a/sympy/physics/quantum/tests/test_tensorproduct.py b/sympy/
2580 physics/quantum/tests/test_tensorproduct.py
2581 2 --- a/sympy/physics/quantum/tests/test_tensorproduct.py
2582 3 +++ b/sympy/physics/quantum/tests/test_tensorproduct.py
2583 4 @@ -47,6 +47,27 @@
2584 5
2585 6     def test_tensor_product_simp():
2586 7         assert tensor_product_simp(TP(A, B)*TP(B, C)) == TP(A*B, B*C)
2587 8 +
2588 9 +         # Test tensor product powers
2589 10 +         a = symbols('a', commutative=False)
2590 11 +         t1 = TP(1, 1) * TP(1, 1)
2591 12 +         t1_simp = tensor_product_simp(t1)
2592 13 +         assert t1_simp == TP(1, 1), f"Expected 1x1, got {t1_simp}"
2593 14 +
2594 15 +         # Test with Pauli matrices
2595 16 +         from sympy.physics.paulialgebra import Pauli

```



```

2592 17 + t2 = TP(1, Pauli(3)) * TP(1, Pauli(3))
2593 18 + t2_simp = tensor_product_simp(t2)
2594 19 + assert t2_simp == TP(1, 1), f"Expected 1x1, got {t2_simp}"
2595 20 +
2596 21 + # Test substitution cases from issue
2597 22 + expr1 = TP(1, 1) * TP(1, a)
2598 23 + result1 = expr1.subs(a, 1)
2599 24 + assert tensor_product_simp(result1) == TP(1, 1), f"Substitution 1
2600 25 + failed: {result1}"
2601 26 + expr2 = TP(1, Pauli(3)) * TP(1, a)
2602 27 + result2 = expr2.subs(a, Pauli(3))
2603 28 + assert tensor_product_simp(result2) == TP(1, 1), f"Substitution 2
2604 29 + failed: {result2}"
2605 30
2606 31 def test_issue_5923():

```

Listing 18: Model predicted test patch of sympy...sympy-13974.

G EMERGENT PARALLEL SCALING

Compared to the self-play method that leverages execution during test time, parallel scaling (Chen et al., 2025b; Pan et al., 2025) is a recently-introduced test-time scaling method that aggregates multiple independent rollouts for additional improvement. While previous works incentivize the ability of parallel scaling through training, or train a separate reward model (Pan et al., 2024; Jain et al., 2025) for patch selection, we observe that the parallel scaling effect emerges in our model *without any specific training*. Specifically, patch candidates from parallel rollouts of BugFixer are used to prompt the model to summarize their differences and compose a new bugfix patch. Listing 19 is the prompt template of parallel scaling. In the emplate, `problem_statement` is the GitHub issue, and `trajs_content` represents the content of multiple patch candidates.

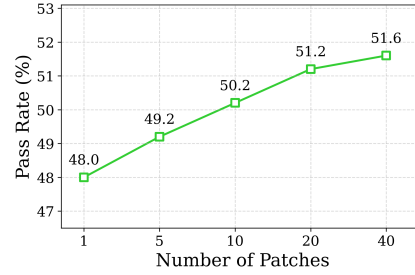


Figure 12: Emergent parallel scaling results on SWE-bench Verified.

```

2626 1232 We are currently solving the following issue within our repository.
2627 1233 Here is the issue text:
2628 1233 --- BEGIN ISSUE ---
2629 1234 {problem_statement}
2630 1235 --- END ISSUE ---
2631 1236
2632 1237 Below are some thinking trajectories, each from llm reasoning model.
2633 1238 Only one trajectory is right.
2634 1238 --- BEGIN FILE ---
2635 1240 {trajs_content}
2636 1241 ---
2637 1242 --- END FILE ---
2638 1243
2639 1244 Please first summary and analyze the key differences between the
2640 1245 trajectories, and then generate *SEARCH/REPLACE* edits to fix the
2641 1246 issue.
2642 1246 Every *SEARCH/REPLACE* edit must use this format:
2643 1247 1. The file path
2644 1248 2. The start of search block: <<<<<< SEARCH
2645 1249 3. A contiguous chunk of lines to search for in the existing source
2646 1250 code
2647 1250 4. The dividing line: =====

```

```

2646 1251 5. The lines to replace into the source code
2647 1252 6. The end of the replace block: >>>>>> REPLACE
2648 1253
2649 1254 Here is an example:
2650 1255
2651 1256 ```python
2652 1257 ### mathweb/flask/app.py
2653 1258 <<<<<<< SEARCH
2654 1259 from flask import Flask
2655 1260 =====
2656 1261 import math
2657 1262 from flask import Flask
2658 1263 >>>>>>> REPLACE
2659 1264 ```
2660 1265
2661 1266 Please note that the *SEARCH/REPLACE* edit REQUIRES PROPER INDENTATION.
2662 1267 If you would like to add the line ' print(x)', you must
2663 1268 fully write that out, with all those spaces before the code!
2664 1269 Wrap the *SEARCH/REPLACE* edit in blocks ```python...```.
2665 1270 The summary of the key differences between the trajectories should be
2666 1271 in the thinking part.

```

Listing 19: The prompt template for parallel scaling.

The results in Figure 12 show that the performance of the parallel aggregation improves as the number of patch candidates in the prompt increases. The advantage of this scaling paradigm over majority voting lies in its ability to leverage the model’s own capacity to analyze multiple candidate patches, thereby surpassing the simplistic approach of weighting answers merely by their frequency of occurrence: Listing 20 covers a full example with 14 different input patch candidates, and how the model successfully recognizes the correct one and generates a novel patch with additional reasoning.

```

2673 1269 Prompt:
2674 1270 We are currently solving the following issue within our repository.
2675 1271 Here is the issue text:
2676 1272 --- BEGIN ISSUE ---
2677 1273 Data <@filename isn't relative to the YAML file
2678 1274 The [docs say](https://gabbi.readthedocs.io/en/latest/format.html#data)
2679 1275 :
2680 1276
2681 1277 > If the value is a string that begins with <@ then the rest of the
2682 1278 string is treated as the name of a file to be loaded from the same
2683 1279 directory as the YAML file.
2684 1280
2685 1281 But I haven't found it works like this unless I cd into the directory
2686 1282 containing the yaml file.
2687 1283
2688 1284 `_load_data_file` says:
2689 1285 ```python
2690 1286 path = os.path.join(self.test_directory, os.path.basename(filename))
2691 1287 ```
2692 1288 this does a few things:
2693 1289 - makes the path begin with `self.test_directory` (defaults to `.` , the
2694 1290 current working directory)
2695 1291 - disregards any folders in the <@ `filename` path
2696 1292 - appends `filename`
2697 1293
2698 1294 This means, if I'm in `./`, and I have a test that says `data: <@cake.
2699 1295 jpg`, I cannot run my tests as advised in the docs:
2700 1296 `gabbi-run -- /my/test.yaml /my/other.yaml`
2701 1297
2702 1298 `FileNotFoundError: [Errno 2] No such file or directory: './cake.jpg`
2703 1299
2704 1300 So in our test running script, we have to start with `cd tests/`.

```

```

2700 1295 My preference, would be to make the path relative to the location of
2701 1296 the yaml file. That way, I can run my tests from whatever directory.
2702 1296
2703 1297 Obviously this isn't possible when redirecting with '<' as the filename
2704 1298 is never seen by gabbi. So I'm happy to discuss other ideas.
2705 1299
2706 1300 ---
2707 1301 Furthermore I can't keep my test data in a child directory, unless I cd
2708 1302 into that directory like:
2709 1303 ```shell
2710 1304 cd yaml_tests/test_data
2711 1305 gabbi-run -- ../test_things.yaml
2712 1306 ```
2713 1307 So for that reason, I'd like to allow directories like '@test_data/
2714 1308 filename.txt' to be included in the path.
2715 1309
2716 1310 ---
2717 1311 Happy to write the patch, if we come up with an approach here.
2718 1312 --- END ISSUE ---
2719 1313
2720 1314 Below are some thinking trajectories, each from llm reasoning model.
2721 1315
2722 1316 --- BEGIN FILE ---
2723 1317 ```
2724 1318 Answer 1:
2725 1319 ```python
2726 1320 ### gabbi/runner.py
2727 1321 <<<<<<< SEARCH
2728 1322 import argparse
2729 1323 from importlib import import_module
2730 1324 import sys
2731 1325 import unittest
2732 1326 =====
2733 1327 import argparse
2734 1328 from importlib import import_module
2735 1329 import os
2736 1330 import sys
2737 1331 import unittest
2738 1332 >>>>>>> REPLACE
2739 1333 ```
2740 1334 ```python
2741 1335 ### gabbi/runner.py
2742 1336 <<<<<<< SEARCH
2743 1337 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2744 1338 False,
2745 1339 failfast=False):
2746 1340 """Run the tests from the YAML in handle."""
2747 1341 data = utils.load_yaml(handle)
2748 1342 if force_ssl:
2749 1343     if 'defaults' in data:
2750 1344         data['defaults']['ssl'] = True
2751 1345     else:
2752 1346         data['defaults'] = {'ssl': True}
2753 1347
2754 1348 loader = unittest.defaultTestLoader
2755 1349 test_suite = suite maker.test_suite_from_dict(
2756 1350     loader, 'input', data, '.', host, port, None, None, prefix=
2757 1351     prefix,
2758 1352     handlers=handler_objects)
2759 1353
2760 1354 result = ConciseTestRunner(

```

```

2754      verbosity=2, failfast=failfast).run(test_suite)
2755      return result.wasSuccessful()
2756  =====
2757  def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2758      False,
2759          failfast=False, test_directory='.'):
2760      """Run the tests from the YAML in handle."""
2761      data = utils.load_yaml(handle)
2762      if force_ssl:
2763          if 'defaults' in data:
2764              data['defaults']['ssl'] = True
2765          else:
2766              data['defaults'] = {'ssl': True}
2767
2768      loader = unittest.defaultTestLoader
2769      test_suite = suite_maker.test_suite_from_dict(
2770          loader, 'input', data, test_directory, host, port, None, None,
2771          prefix=prefix, handlers=handler_objects)
2772
2773      result = ConciseTestRunner(
2774          verbosity=2, failfast=failfast).run(test_suite)
2775      return result.wasSuccessful()
2776  >>>>>> REPLACE
2777  ```
2778  ```python
2779  ### gabbi/runner.py
2780  <<<<<<< SEARCH
2781  else:
2782      for input_file in input_files:
2783          with open(input_file, 'r') as fh:
2784              success = run_suite(fh, handler_objects, host, port,
2785                  prefix, force_ssl, failfast)
2786  =====
2787  else:
2788      for input_file in input_files:
2789          test_directory = os.path.dirname(input_file) or '.'
2790          with open(input_file, 'r') as fh:
2791              success = run_suite(fh, handler_objects, host, port,
2792                  prefix, force_ssl, failfast,
2793                  test_directory=test_directory)
2794  >>>>>> REPLACE
2795  ```
2796
2797  Answer 2:
2798  ```python
2799  ### gabbi/runner.py
2800  <<<<<<< SEARCH
2801  def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2802      False,
2803          failfast=False):
2804      =====
2805  def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2806      False,
2807          failfast=False, test_directory='.'):
2808  >>>>>> REPLACE
2809  ```
2810  ```python
2811  ### gabbi/runner.py
2812  <<<<<<< SEARCH
2813      test_suite = suite_maker.test_suite_from_dict(
2814          loader, 'input', data, '.', host, port, None, None, prefix=
2815          prefix,
2816          handlers=handler_objects)
2817  =====
2818      test_suite = suite_maker.test_suite_from_dict(

```

```

2808 loader, 'input', data, test_directory, host, port, None, None,
2809     prefix=prefix,
2810     handlers=handler_objects)
2811 >>>>>> REPLACE
2812 ```
2813 ```python
2814 """ gabbi/runner.py
2815 <<<<<<< SEARCH
2816     success = run_suite(fh, handler_objects, host, port,
2817         prefix, force_ssl, failfast)
2818 =====
2819     test_directory = os.path.dirname(input_file) or '.'
2820     success = run_suite(fh, handler_objects, host, port,
2821         prefix, force_ssl, failfast,
2822         test_directory=test_directory)
2823 >>>>>> REPLACE
2824 ```
2825
2826 Answer 3:
2827 ```python
2828 """ gabbi/runner.py
2829 <<<<<<< SEARCH
2830 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2831     False,
2832     failfast=False):
2833     """Run the tests from the YAML in handle."""
2834     data = utils.load_yaml(handle)
2835     if force_ssl:
2836         if 'defaults' in data:
2837             data['defaults']['ssl'] = True
2838         else:
2839             data['defaults'] = {'ssl': True}
2840
2841     loader = unittest.defaultTestLoader
2842     test_suite = suite_maker.test_suite_from_dict(
2843         loader, 'input', data, '.', host, port, None, None, prefix=
2844         prefix,
2845         handlers=handler_objects)
2846
2847     result = ConciseTestRunner(
2848         verbosity=2, failfast=failfast).run(test_suite)
2849     return result.wasSuccessful()
2850 =====
2851 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2852     False,
2853     failfast=False, test_directory='.'):
2854     """Run the tests from the YAML in handle."""
2855     data = utils.load_yaml(handle)
2856     if force_ssl:
2857         if 'defaults' in data:
2858             data['defaults']['ssl'] = True
2859         else:
2860             data['defaults'] = {'ssl': True}
2861
2862     loader = unittest.defaultTestLoader
2863     test_suite = suite_maker.test_suite_from_dict(
2864         loader, 'input', data, test_directory, host, port, None, None,
2865         prefix=prefix, handlers=handler_objects)
2866
2867     result = ConciseTestRunner(
2868         verbosity=2, failfast=failfast).run(test_suite)
2869     return result.wasSuccessful()
2870 >>>>>> REPLACE
2871 ```
2872 ```python

```

```

2862 1475 ### gabbi/runner.py
2863 1476 <<<<<<< SEARCH
2864 1477     for input_file in input_files:
2865 1478         with open(input_file, 'r') as fh:
2866 1479             success = run_suite(fh, handler_objects, host, port,
2867 1480                             prefix, force_ssl, failfast)
2868 1481             if not failure: # once failed, this is considered
2869 1482                 immutable
2870 1483                 failure = not success
2871 1484             if failure and failfast:
2872 1485                 break
2873 1486
2874 1487     for input_file in input_files:
2875 1488         test_directory = os.path.dirname(input_file)
2876 1489         with open(input_file, 'r') as fh:
2877 1490             success = run_suite(
2878 1491                 fh, handler_objects, host, port, prefix, force_ssl,
2879 1492                 failfast, test_directory=test_directory)
2880 1493             if not failure: # once failed, this is considered
2881 1494                 immutable
2882 1495                 failure = not success
2883 1496             if failure and failfast:
2884 1497                 break
2885 1498
2886 1499 >>>>>> REPLACE
2887 1500 '''
2888 1501
2889 1502 Answer 4:
2890 1503 ```python
2891 1504 ### gabbi/runner.py
2892 1505 <<<<<<< SEARCH
2893 1506 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2894 1507     False,
2895 1508                 failfast=False):
2896 1509     =====
2897 1510     def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2898 1511         False,
2899 1512                 failfast=False, test_directory='.'):
2900 1513         >>>>>> REPLACE
2901 1514         '''
2902 1515         ```python
2903 1516         ### gabbi/runner.py
2904 1517         <<<<<<< SEARCH
2905 1518             test_suite = suitemaker.test_suite_from_dict(
2906 1519                 loader, 'input', data, '.', host, port, None, None, prefix=
2907 1520                 prefix,
2908 1521                 handlers=handler_objects)
2909 1522         =====
2910 1523             test_suite = suitemaker.test_suite_from_dict(
2911 1524                 loader, 'input', data, test_directory, host, port, None, None,
2912 1525                 prefix=prefix, handlers=handler_objects)
2913 1526         >>>>>> REPLACE
2914 1527         '''
2915 1528         ```python
2916 1529         ### gabbi/runner.py
2917 1530         <<<<<<< SEARCH
2918 1531             for input_file in input_files:
2919 1532                 with open(input_file, 'r') as fh:
2920 1533                     success = run_suite(fh, handler_objects, host, port,

```

```

2916      success = run_suite(fh, handler_objects, host, port,
1534
2917      prefix, force_ssl, failfast,
1535
2918      test_directory=test_directory)
1536
2919 >>>>>> REPLACE
1537
2920 ```
1538
2921 Answer 5:
1539
2922 ```python
1540
2923 ### gabbi/runner.py
1541
2924 <<<<<<< SEARCH
1542
2925 import sys
1543
2926 import unittest
1544
2927 =====
1545
2928 import os
1546
2929 import sys
1547
2930 import unittest
1548
2931 >>>>>> REPLACE
1549
2932 ```
1550
2933 ```python
1551
2934 ### gabbi/runner.py
1552
2935 <<<<<<< SEARCH
1553
2936 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
1554
2937     False,
1555
2938     failfast=False):
1556
2939     =====
1557
2940 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
1558
2941     False,
1559
2942     failfast=False, test_directory='.'):
1560
2943 >>>>>> REPLACE
1561
2944 ```
1562
2945 ```python
1563
2946 ### gabbi/runner.py
1564
2947 <<<<<<< SEARCH
1565
2948     test_suite = suitemaker.test_suite_from_dict(
1566
2949         loader, 'input', data, '.', host, port, None, None, prefix=
1567
2950         prefix,
1568
2951         handlers=handler_objects)
1569
2952     =====
1570
2953     test_suite = suitemaker.test_suite_from_dict(
1571
2954         loader, 'input', data, test_directory, host, port, None, None,
1572
2955         prefix=prefix, handlers=handler_objects)
1573
2956 >>>>>> REPLACE
1574
2957 ```
1575
2958 ```python
1576
2959 ### gabbi/runner.py
1577
2960 <<<<<<< SEARCH
1578
2961     for input_file in input_files:
1579
2962         with open(input_file, 'r') as fh:
1580
2963             success = run_suite(fh, handler_objects, host, port,
1581
2964                 prefix, force_ssl, failfast)
1582
2965     =====
1583
2966     for input_file in input_files:
1584
2967         test_directory = os.path.dirname(input_file) or '.'
1585
2968         with open(input_file, 'r') as fh:
1586
2969             success = run_suite(fh, handler_objects, host, port,
1587
2970                 prefix, force_ssl, failfast,
1588
2971                 test_directory=test_directory)
1589
2972 >>>>>> REPLACE
1590
2973 ```
1591
2974 Answer 6:
1592
2975 ```python
1593
2976 ### gabbi/driver.py
1594
2977 <<<<<<< SEARCH
1595
2978     file_suite = suitemaker.test_suite_from_dict(

```

```

2970 loader, test_base_name, suite_dict, path, host, port,
2971 fixture_module, intercept, prefix=prefix,
2972 test_loader_name=test_loader_name, handlers=handler_objects
2973 ,
2974 inner_fixtures=inner_fixtures)
2975 =====
2976 test_file_dir = os.path.dirname(test_file)
2977 file_suite = suitemaker.test_suite_from_dict(
2978 loader, test_base_name, suite_dict, test_file_dir, host,
2979 port,
2980 fixture_module, intercept, prefix=prefix,
2981 test_loader_name=test_loader_name, handlers=handler_objects
2982 ,
2983 inner_fixtures=inner_fixtures)
2984 >>>>>> REPLACE
2985 '''
2986 ```python
2987 ### gabbi/runner.py
2988 <<<<<<< SEARCH
2989 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2990 False,
2991 failfast=False):
2992 =====
2993 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2994 False,
2995 failfast=False, test_directory='.'):
2996 >>>>>> REPLACE
2997 '''
2998 ```python
2999 ### gabbi/runner.py
3000 <<<<<<< SEARCH
3001 test_suite = suitemaker.test_suite_from_dict(
3002 loader, 'input', data, '.', host, port, None, None, prefix=
3003 prefix,
3004 handlers=handler_objects)
3005 =====
3006 test_suite = suitemaker.test_suite_from_dict(
3007 loader, 'input', data, test_directory, host, port, None, None,
3008 prefix=prefix, handlers=handler_objects)
3009 >>>>>> REPLACE
3010 '''
3011 ```python
3012 ### gabbi/runner.py
3013 <<<<<<< SEARCH
3014 success = run_suite(fh, handler_objects, host, port,
3015 prefix, force_ssl, failfast)
3016 =====
3017 test_directory = os.path.dirname(input_file)
3018 success = run_suite(fh, handler_objects, host, port,
3019 prefix, force_ssl, failfast,
3020 test_directory)
3021 >>>>>> REPLACE
3022 '''
3023
Answer 7:
3024 ```python
3025 ### gabbi/runner.py
3026 <<<<<<< SEARCH
3027 import argparse
3028 from importlib import import_module
3029 import sys
3030 import unittest
3031 =====
3032 import argparse
3033 from importlib import import_module

```



```

3024 1654 import os
3025 1655 import sys
3026 1656 import unittest
3027 1657 >>>>>> REPLACE
3028 1658 ```
3029 1659 ```python
3030 1660 ### gabbi/runner.py
3031 1661 <<<<<<< SEARCH
3032 1662 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3033 1663     False,
3034 1664         failfast=False):
3035 1665     """Run the tests from the YAML in handle."""
3036 1666     data = utils.load_yaml(handle)
3037 1667     if force_ssl:
3038 1668         if 'defaults' in data:
3039 1669             data['defaults']['ssl'] = True
3040 1670         else:
3041 1671             data['defaults'] = {'ssl': True}
3042 1672     loader = unittest.defaultTestLoader
3043 1673     test_suite = suite_maker.test_suite_from_dict(
3044 1674         loader, 'input', data, '.', host, port, None, None, prefix=
3045 1675         prefix,
3046 1676         handlers=handler_objects)
3047 1677 =====
3048 1678 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3049 1679     False,
3050 1680         failfast=False, test_directory='.'):
3051 1681     """Run the tests from the YAML in handle."""
3052 1682     data = utils.load_yaml(handle)
3053 1683     if force_ssl:
3054 1684         if 'defaults' in data:
3055 1685             data['defaults']['ssl'] = True
3056 1686         else:
3057 1687             data['defaults'] = {'ssl': True}
3058 1688     loader = unittest.defaultTestLoader
3059 1689     test_suite = suite_maker.test_suite_from_dict(
3060 1690         loader, 'input', data, test_directory, host, port, None, None,
3061 1691         prefix=prefix, handlers=handler_objects)
3062 1692 >>>>>> REPLACE
3063 1693 ```
3064 1694 ```python
3065 1695 ### gabbi/runner.py
3066 1696 <<<<<<< SEARCH
3067 1697     with open(input_file, 'r') as fh:
3068 1698         success = run_suite(fh, handler_objects, host, port,
3069 1699             prefix, force_ssl, failfast)
3070 1700         if not failure: # once failed, this is considered
3071 1701             immutable
3072 1702             failure = not success
3073 1703         if failure and failfast:
3074 1704             break
3075 1705 =====
3076 1706     test_dir = os.path.dirname(os.path.abspath(input_file))
3077 1707     with open(input_file, 'r') as fh:
3078 1708         success = run_suite(fh, handler_objects, host, port,
3079 1709             prefix, force_ssl, failfast,
3080 1710             test_directory=test_dir)
3081 1711         if not failure: # once failed, this is considered
3082 1712             immutable
3083 1713             failure = not success
3084 1714         if failure and failfast:
3085 1715             break
3086 1716 >>>>>> REPLACE

```

```

3078 1713 '''
3079 1714
3080 1715 Answer 8:
3081 1716 ```python
3082 1717 ### gabbi/runner.py
3083 1718 <<<<<<< SEARCH
3084 1719     data = utils.load_yaml(handle)
3085 1720     if force_ssl:
3086 1721         if 'defaults' in data:
3087 1722             data['defaults']['ssl'] = True
3088 1723         else:
3089 1724             data['defaults'] = {'ssl': True}
3090 1725
3091 1726     loader = unittest.defaultTestLoader
3092 1727     test_suite = suitemaker.test_suite_from_dict(
3093 1728         loader, 'input', data, '.', host, port, None, None, prefix=
3094 1729         prefix,
3095 1730         handlers=handler_objects)
3096 1731
3097 1732     data = utils.load_yaml(handle)
3098 1733     if force_ssl:
3099 1734         if 'defaults' in data:
3100 1735             data['defaults']['ssl'] = True
3101 1736         else:
3102 1737             data['defaults'] = {'ssl': True}
3103 1738
3104 1739     # Determine test directory based on input handle
3105 1740     if hasattr(handle, 'name'):
3106 1741         test_directory = os.path.dirname(os.path.abspath(handle.name))
3107 1742     else:
3108 1743         test_directory = '.'
3109 1744
3110 1745     loader = unittest.defaultTestLoader
3111 1746     test_suite = suitemaker.test_suite_from_dict(
3112 1747         loader, 'input', data, test_directory, host, port, None, None,
3113 1748         prefix=prefix, handlers=handler_objects)
3114 1749 >>>>>> REPLACE
3115 1750 '''
3116 1751
3117 1752 Answer 9:
3118 1753 ```python
3119 1754 ### gabbi/driver.py
3120 1755 <<<<<<< SEARCH
3121 1756     for test_file in glob.iglob('%s/*.yaml' % path):
3122 1757         if '_' in os.path.basename(test_file):
3123 1758             warnings.warn(exception.GabbiSyntaxWarning(
3124 1759                 "'_' in test filename %s. This can break suite grouping
3125 1760                 ."
3126 1761                 % test_file))
3127 1762         if intercept:
3128 1763             host = str(uuid.uuid4())
3129 1764             suite_dict = utils.load_yaml(yaml_file=test_file)
3130 1765             test_base_name = os.path.splitext(os.path.basename(test_file))
3131 1766             [0]
3132 1767             if all_test_base_name:
3133 1768                 test_base_name = '%s%s' % (all_test_base_name,
3134 1769                 test_base_name)
3135 1770
3136 1771         if require_ssl:
3137 1772             if 'defaults' in suite_dict:
3138 1773                 suite_dict['defaults']['ssl'] = True
3139 1774             else:
3140 1775                 suite_dict['defaults'] = {'ssl': True}
3141 1776
3142 1777     file_suite = suitemaker.test_suite_from_dict(

```

```

3132 loader, test_base_name, suite_dict, path, host, port,
3133 fixture_module, intercept, prefix=prefix,
3134 test_loader_name=test_loader_name, handlers=handler_objects
3135 ,
3136 inner_fixtures=inner_fixtures)
3137 =====
3138 for test_file in glob.iglob('%s/*.yaml' % path):
3139     if '_' in os.path.basename(test_file):
3140         warnings.warn(exception.GabbiSyntaxWarning(
3141             "'_' in test filename %s. This can break suite grouping
3142             ."
3143             % test_file))
3144     if intercept:
3145         host = str(uuid.uuid4())
3146         suite_dict = utils.load_yaml(yaml_file=test_file)
3147         test_base_name = os.path.splitext(os.path.basename(test_file))
3148         [0]
3149         if all_test_base_name:
3150             test_base_name = '%s_%s' % (all_test_base_name,
3151                 test_base_name)
3152     if require_ssl:
3153         if 'defaults' in suite_dict:
3154             suite_dict['defaults']['ssl'] = True
3155         else:
3156             suite_dict['defaults'] = {'ssl': True}
3157     # Use absolute path of the YAML file's directory
3158     test_file_dir = os.path.dirname(os.path.abspath(test_file))
3159     file_suite = suitemaker.test_suite_from_dict(
3160         loader, test_base_name, suite_dict, test_file_dir, host,
3161         port,
3162         fixture_module, intercept, prefix=prefix,
3163         test_loader_name=test_loader_name, handlers=handler_objects
3164         ,
3165         inner_fixtures=inner_fixtures)
3166 >>>>>> REPLACE
3167 ```
3168
3169 Answer 10:
3170 ```python
3171 ### gabbi/runner.py
3172 <<<<<<< SEARCH
3173 import argparse
3174 from importlib import import_module
3175 import sys
3176 import unittest
3177 =====
3178 import argparse
3179 from importlib import import_module
3180 import os
3181 import sys
3182 import unittest
3183 >>>>>> REPLACE
3184 ```
3185 ```python
3186 ### gabbi/runner.py
3187 <<<<<<< SEARCH
3188 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3189     False,
3190             failfast=False):
3191     =====
3192 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3193     False,
3194             failfast=False, test_directory='.'):

```

```

3186 1831 >>>>>> REPLACE
3187 1832 ```
3188 1833 ```python
3189 1834 ### gabbi/runner.py
3190 1835 <<<<<<< SEARCH
3191 1836     test_suite = suitemaker.test_suite_from_dict(
3192 1837         loader, 'input', data, '.', host, port, None, None, prefix=
3193 1838         prefix,
3194 1839         force_ssl=force_ssl, failfast=failfast,
3195 1840         handlers=handler_objects)
3196 1841     test_suite = suitemaker.test_suite_from_dict(
3197 1842         loader, 'input', data, test_directory, host, port, None, None,
3198 1843         prefix=prefix, force_ssl=force_ssl, failfast=failfast,
3199 1844         handlers=handler_objects)
3200 1845 >>>>>> REPLACE
3201 1846 ```
3202 1847 ```python
3203 1848 ### gabbi/runner.py
3204 1849 <<<<<<< SEARCH
3205 1850         success = run_suite(fh, handler_objects, host, port,
3206 1851         prefix, force_ssl, failfast)
3207 1852     test_dir = os.path.dirname(input_file)
3208 1853     success = run_suite(fh, handler_objects, host, port,
3209 1854         prefix, force_ssl, failfast,
3210 1855         test_directory=test_dir)
3211 1856 >>>>>> REPLACE
3212 1857 ```
3213 1858 ```python
3214 1859 ### gabbi/case.py
3215 1860 <<<<<<< SEARCH
3216 1861     path = os.path.join(self.test_directory, os.path.basename(filename)
3217 1862     )
3218 1863     path = os.path.join(self.test_directory, filename)
3219 1864 >>>>>> REPLACE
3220 1865 ```
3221 1866
3222 1867 Answer 11:
3223 1868 ```python
3224 1869 ### gabbi/runner.py
3225 1870 <<<<<<< SEARCH
3226 1871 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3227 1872     False,
3228 1873         failfast=False):
3229 1874     """Run the tests from the YAML in handle."""
3230 1875     data = utils.load_yaml(handle)
3231 1876     if force_ssl:
3232 1877         if 'defaults' in data:
3233 1878             data['defaults']['ssl'] = True
3234 1879         else:
3235 1880             data['defaults'] = {'ssl': True}
3236 1881     loader = unittest.defaultTestLoader
3237 1882     test_suite = suitemaker.test_suite_from_dict(
3238 1883         loader, 'input', data, '.', host, port, None, None, prefix=
3239 1884         prefix,
3240 1885         handlers=handler_objects)
3241 1886     def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3242 1887         False,
3243 1888             failfast=False, test_directory='.'):
3244 1889         """Run the tests from the YAML in handle."""
3245 1890         data = utils.load_yaml(handle)

```

```

3240 1891     if force_ssl:
3241 1892         if 'defaults' in data:
3242 1893             data['defaults']['ssl'] = True
3243 1894         else:
3244 1895             data['defaults'] = {'ssl': True}
3245 1896
3246 1897     loader = unittest.defaultTestLoader
3247 1898     test_suite = suitemaker.test_suite_from_dict(
3248 1899         loader, 'input', data, test_directory, host, port,
3249 1900         None, None, prefix=prefix, handlers=handler_objects)
3250 1901 >>>>>> REPLACE
3251 1902 '''
3252 1903 '''python
3253 1904 ### gabbi/runner.py
3254 1905 <<<<<<< SEARCH
3255 1906         success = run_suite(fh, handler_objects, host, port,
3256 1907                             prefix, force_ssl, failfast)
3257 1908
3258 1909         test_dir = os.path.dirname(input_file)
3259 1910         success = run_suite(fh, handler_objects, host, port,
3260 1911                             prefix, force_ssl, failfast,
3261 1912                             test_directory=test_dir)
3262 1913 >>>>>> REPLACE
3263 1914 '''
3264 1915
3265 1916 Answer 12:
3266 1917 '''python
3267 1918 ### gabbi/runner.py
3268 1919 <<<<<<< SEARCH
3269 1920 import argparse
3270 1921 from importlib import import_module
3271 1922 import sys
3272 1923 import unittest
3273 1924
3274 1925
3275 1926 import argparse
3276 1927 from importlib import import_module
3277 1928 import os
3278 1929 import sys
3279 1930 import unittest
3280 1931 >>>>>> REPLACE
3281 1932 '''
3282 1933 '''python
3283 1934 ### gabbi/runner.py
3284 1935 <<<<<<< SEARCH
3285 1936     else:
3286 1937         for input_file in input_files:
3287 1938             with open(input_file, 'r') as fh:
3288 1939                 success = run_suite(fh, handler_objects, host, port,
3289 1940                                     prefix, force_ssl, failfast)
3290 1941
3291 1942         test_dir = os.path.dirname(os.path.abspath(input_file))
3292 1943         with open(input_file, 'r') as fh:
3293 1944             success = run_suite(fh, handler_objects, host, port,
3294 1945                                 prefix, force_ssl, failfast,
3295 1946                                 test_directory=test_dir)
3296 1947 >>>>>> REPLACE
3297 1948 '''
3298 1949 '''python
3299 1950 ### gabbi/runner.py
3300 1951 <<<<<<< SEARCH
3301 1952 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3302 1953     False,
3303 1954     failfast=False):

```

```

3294 1955 """Run the tests from the YAML in handle."""
3295 1956 data = utils.load_yaml(handle)
3296 1957 if force_ssl:
3297 1958     if 'defaults' in data:
3298 1959         data['defaults']['ssl'] = True
3299 1960     else:
3300 1961         data['defaults'] = {'ssl': True}
3301 1962
3302 1963 loader = unittest.defaultTestLoader
3303 1964 test_suite = suite_maker.test_suite_from_dict(
3304 1965     loader, 'input', data, '.', host, port, None, None, prefix=
3305 1966     prefix,
3306 1967     handlers=handler_objects)
3307 1968
3308 1969 =====
3309 1970 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3310 1971     False,
3311 1972         failfast=False, test_directory='.'):
3312 1973     """Run the tests from the YAML in handle."""
3313 1974     data = utils.load_yaml(handle)
3314 1975     if force_ssl:
3315 1976         if 'defaults' in data:
3316 1977             data['defaults']['ssl'] = True
3317 1978         else:
3318 1979             data['defaults'] = {'ssl': True}
3319 1980
3320 1981 loader = unittest.defaultTestLoader
3321 1982 test_suite = suite_maker.test_suite_from_dict(
3322 1983     loader, 'input', data, test_directory, host, port, None, None,
3323 1984     prefix=prefix, handlers=handler_objects)
3324 1985 >>>>>> REPLACE
3325 1986 ```
3326 1987
3327 1988 Answer 13:
3328 1989 ```python
3329 1990 ### gabbi/runner.py
3330 1991 <<<<<<< SEARCH
3331 1992 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3332 1993     False,
3333 1994         failfast=False):
3334 1995     """Run the tests from the YAML in handle."""
3335 1996     data = utils.load_yaml(handle)
3336 1997     if force_ssl:
3337 1998         if 'defaults' in data:
3338 1999             data['defaults']['ssl'] = True
3339 2000         else:
3340 2001             data['defaults'] = {'ssl': True}
3341 2002
3342 2003 loader = unittest.defaultTestLoader
3343 2004 test_suite = suite_maker.test_suite_from_dict(
3344 2005     loader, 'input', data, '.', host, port, None, None, prefix=
3345 2006     prefix,
3346 2007     handlers=handler_objects)
3347 2008
3348 2009 =====
3349 2010 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3350 2011     False,
3351 2012         failfast=False, test_directory='.'):
3352 2013     """Run the tests from the YAML in handle."""
3353 2014     data = utils.load_yaml(handle)
3354 2015     if force_ssl:
3355 2016         if 'defaults' in data:
3356 2017             data['defaults']['ssl'] = True
3357 2018         else:
3358 2019             data['defaults'] = {'ssl': True}
3359 2020
3360 2021 loader = unittest.defaultTestLoader

```

```

3348 2015 test_suite = suitemaker.test_suite_from_dict(
3349 2016     loader, 'input', data, test_directory, host, port, None, None,
3350 2017     prefix=prefix, handlers=handler_objects)
3351 2018 >>>>>> REPLACE
3352 2019 ```
3353 2020 ```python
3354 2021 ### gabbi/runner.py
3355 2022 <<<<<<< SEARCH
3356 2023     for input_file in input_files:
3357 2024         with open(input_file, 'r') as fh:
3358 2025             success = run_suite(fh, handler_objects, host, port,
3359 2026                             prefix, force_ssl, failfast)
3360 2027
3361 2028     for input_file in input_files:
3362 2029         test_dir = os.path.dirname(os.path.abspath(input_file))
3363 2030         with open(input_file, 'r') as fh:
3364 2031             success = run_suite(fh, handler_objects, host, port,
3365 2032                             prefix, force_ssl, failfast,
3366 2033                             test_directory=test_dir)
3367 2034
3368 2035 >>>>>> REPLACE
3369 2036 ```
3370 2037
3371 2038 Answer 14:
3372 2039 ```python
3373 2040 ### gabbi/runner.py
3374 2041 <<<<<<< SEARCH
3375 2042     for input_file in input_files:
3376 2043         with open(input_file, 'r') as fh:
3377 2044             success = run_suite(fh, handler_objects, host, port,
3378 2045                             prefix, force_ssl, failfast)
3379 2046
3380 2047     for input_file in input_files:
3381 2048         with open(input_file, 'r') as fh:
3382 2049             success = run_suite(input_file, fh, handler_objects,
3383 2050                             host, port,
3384 2051                             prefix, force_ssl, failfast)
3385 2052
3386 2053 >>>>>> REPLACE
3387 2054 ```
3388 2055 ```python
3389 2056 ### gabbi/runner.py
3390 2057 <<<<<<< SEARCH
3391 2058 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3392 2059     False,
3393 2060     failfast=False):
3394 2061     """Run the tests from the YAML in handle."""
3395 2062     data = utils.load_yaml(handle)
3396 2063     if force_ssl:
3397 2064         if 'defaults' in data:
3398 2065             data['defaults']['ssl'] = True
3399 2066         else:
3400 2067             data['defaults'] = {'ssl': True}
3401 2068
3402 2069     loader = unittest.defaultTestLoader
3403 2070     test_directory = '.'
3404 2071     test_suite = suitemaker.test_suite_from_dict(
3405 2072         loader, 'input', data, test_directory, host, port, None, None,
3406 2073         prefix=prefix,
3407 2074         handlers=handler_objects)
3408 2075
3409 2076     def run_suite(input_file, handle, handler_objects, host, port, prefix,
3410 2077         force_ssl=False,
3411 2078         failfast=False):
3412 2079         """Run the tests from the YAML in handle."""
3413 2080         data = utils.load_yaml(handle)
3414 2081         if force_ssl:

```

```

3402         if 'defaults' in data:
3403             data['defaults']['ssl'] = True
3404         else:
3405             data['defaults'] = {'ssl': True}
3406
3407         loader = unittest.defaultTestLoader
3408         test_directory = os.path.dirname(input_file) if input_file else '.'
3409         test_suite = suitemaker.test_suite_from_dict(
3410             loader, 'input', data, test_directory, host, port, None, None,
3411             prefix=prefix,
3412             handlers=handler_objects)
3413
3414     >>>>>> REPLACE
3415     ```python
3416     ### gabbi/runner.py
3417     <<<<<<< SEARCH
3418         if not input_files:
3419             success = run_suite(sys.stdin, handler_objects, host, port,
3420                                 prefix, force_ssl, failfast)
3421         =====
3422         if not input_files:
3423             success = run_suite(None, sys.stdin, handler_objects, host,
3424                                 port,
3425                                 prefix, force_ssl, failfast)
3426     >>>>>> REPLACE
3427     ```
3428     --- END FILE ---

```

Please first summary and analyze the key differences between the trajectories, and then generate *SEARCH/REPLACE* edits to fix the issue.

Every *SEARCH/REPLACE* edit must use this format:

1. The file path
2. The start of search block: <<<<<<< SEARCH
3. A contiguous chunk of lines to search for in the existing source code
4. The dividing line: =====
5. The lines to replace into the source code
6. The end of the replace block: >>>>>>> REPLACE

Here is an example:

```

3439     ```python
3440     ### mathweb/flask/app.py
3441     <<<<<<< SEARCH
3442     from flask import Flask
3443     =====
3444     import math
3445     from flask import Flask
3446     >>>>>>> REPLACE
3447     ```

```

Please note that the *SEARCH/REPLACE* edit REQUIRES PROPER INDENTATION. If you would like to add the line ' print(x)', you must fully write that out, with all those spaces before the code! Wrap the *SEARCH/REPLACE* edit in blocks ```python...```. The summary of the key differences between the trajectories should be in the thinking part.

"

Response:

<think>

3456²¹³³ Okay, let's tackle this problem. The user is having an issue where when
 3457 they specify a data file in their YAML test using the <@ syntax,
 3458 the path isn't resolved relative to the YAML file's location.
 3459 Instead, it's using the current working directory. They want the
 3460 path to be relative to the YAML file so they can run tests from any
 3461 directory.

3462²¹³⁴
 3463²¹³⁵ First, I need to understand how the current code is handling the data
 3464 file paths. The user mentioned that in `_load_data_file`, the code
 3465 uses `os.path.join(self.test_directory, os.path.basename(filename))`.
 3466 This takes the `test_directory` (which defaults to `'.'`), discards any
 3467 directories in the filename, and appends the basename. That's why if
 3468 the YAML is in `/my/tests/test.yaml` and the data is `<@data.txt`, it
 3469 looks for `./data.txt` instead of `/my/tests/data.txt`.

3470²¹³⁶
 3471²¹³⁷ The key here is to make sure that the `test_directory` is set to the
 3472 directory where the YAML file is located. Then, when joining with
 3473 the filename (without basename), it would correctly resolve the
 3474 relative path.

3475²¹³⁸
 3476²¹³⁹ Looking at the provided code snippets, the main changes are in `runner.py`
 3477 where the `test_directory` is determined. The user's proposed
 3478 solutions involve modifying the `run_suite` function to accept a
 3479 `test_directory` parameter, which is derived from the input file's
 3480 directory.

3481²¹⁴⁰
 3482²¹⁴¹ In the original code, when running a suite from an input file, the
 3483 `test_directory` is hardcoded to `'.'`. The various answers modify this
 3484 by passing the directory of the input file as `test_directory`. For
 3485 example, in Answer 10, they add the `test_directory` parameter to
 3486 `run_suite` and set it to `os.path.dirname(input_file)`. Then, in `case.py`,
 3487 they change the path joining to use the filename directly
 3488 instead of basename.

3489²¹⁴²
 3490²¹⁴³ Wait, the critical part is in `case.py`'s `_load_data_file` method. The
 3491 original code uses `os.path.basename(filename)`, which strips any
 3492 directories from the filename. So if the YAML has `<@test_data/file.txt`,
 3493 the basename is `file.txt`, and it's joined with `test_directory`.
 3494 But if `test_directory` is the YAML's directory, then it should be
 3495 `YAML_DIR/test_data/file.txt`. However, the current code would do
 3496 `test_directory + file.txt`, which would be `YAML_DIR/file.txt`. That's
 3497 incorrect. So the problem is that the code is taking the basename,
 3498 thus ignoring any subdirectories in the filename.

3499²¹⁴⁴
 3500²¹⁴⁵ Ah, so the fix here is to not take the basename. Instead, the filename
 3501 after `<@` should be treated as a relative path from the YAML's
 3502 directory. So in `case.py`'s `_load_data_file`, instead of using `os.path`
 3503 `.basename(filename)`, we should use the filename as-is. Then, `os.path`
 3504 `.join(self.test_directory, filename)` would correctly resolve the
 3505 path relative to the YAML's directory.

3506²¹⁴⁶
 3507²¹⁴⁷ But wait, the filename could be an absolute path. For example, if the
 3508 YAML has `<@/absolute/path/file.txt`, then `os.path.join` would just use
 3509 the absolute path. However, the documentation says that the
 3510 filename is relative to the YAML file. So absolute paths might not
 3511 be intended. But the code should handle that.

3512²¹⁴⁸
 3513²¹⁴⁹ So combining two changes: in `runner.py`, set `test_directory` to the YAML
 3514 file's directory, and in `case.py`, stop using `basename` and use the
 3515 full filename.

3516²¹⁵⁰
 3517²¹⁵¹ Looking at Answer 10, there's a change in `case.py` from `os.path.basename`
 3518 (`filename`) to `filename`. That's exactly the fix needed. Then, in
 3519 `runner.py`, when running the suite, pass the `test_directory` as the
 3520 directory of the input file.

3510
3511²¹⁵² So the key files to modify are:
3512²¹⁵³
3513²¹⁵⁴ 1. gabbi/runner.py: When calling run_suite, pass test_directory as the
3514²¹⁵⁵ directory of the input file. This involves modifying the run_suite
3515²¹⁵⁶ function to accept test_directory as a parameter and then passing it
3516²¹⁵⁷ when the input file is processed.
3517²¹⁵⁸ 2. gabbi/case.py: Change the path construction from using os.path.
3518²¹⁵⁹ basename(filename) to using the filename directly. This allows
3519²¹⁶⁰ subdirectories in the filename to be included in the path.
3520²¹⁶¹ Looking through the provided code snippets, the one that does both is
3521²¹⁶² Answer 10. Let me check:
3522²¹⁶³
3523²¹⁶⁴ In Answer 10, the runner.py changes add the test_directory parameter
3524²¹⁶⁵ and set it to os.path.dirname(input_file). Then, in case.py, the
3525²¹⁶⁶ path is changed to os.path.join(self.test_directory, filename). That
3526²¹⁶⁷ would correctly resolve the path relative to the YAML's directory,
3527²¹⁶⁸ including any subdirectories in the filename.
3528²¹⁶⁹ But wait, the original code in case.py uses os.path.basename(filename).
3529²¹⁷⁰ So changing that to filename means that if the input is 'test_data/
3530²¹⁷¹ file.txt', it's joined as test_directory/test_data/file.txt. If
3531²¹⁷² test_directory is the YAML's directory, that's correct.
3532²¹⁷³ ----- Omit some reasoning content -----
3533²¹⁷⁴
3534²¹⁷⁵ But perhaps better to use os.path.abspath and then dirname to handle
3535²¹⁷⁶ relative paths correctly. But maybe the existing code already works.
3536²¹⁷⁷ For example, if the input_file is 'tests/test.yaml', then os.path.
3537²¹⁷⁸ dirname would return 'tests', which when joined with os.getcwd()
3538²¹⁷⁹ would be correct. Wait, no. os.path.dirname just gives the directory
3539²¹⁸⁰ part of the path. So if input_file is a relative path, like 'tests/
3540²¹⁸¹ test.yaml', then test_directory would be 'tests', and when os.path.
3541²¹⁸² join is called, it's relative to the current working directory. But
3542²¹⁸³ the code in suitemaker.test_suite_from_dict might expect an absolute
3543²¹⁸⁴ path. However, the original code uses '.' as the test_directory,
3544²¹⁸⁵ which is relative. So perhaps using the directory of the input file
3545²¹⁸⁶ as test_directory is sufficient.
3546²¹⁸⁷ Another thing to consider: when the YAML is loaded from stdin (like via
3547²¹⁸⁸ redirect), the test_directory remains '.'. That's acceptable, as
3548²¹⁸⁹ the documentation mentions that when using <, the filename isn't
3549²¹⁹⁰ seen by Gabbi, so it's not possible to resolve the path relative to
3550²¹⁹¹ the YAML's original location. So in that case, the test_directory
3551²¹⁹² defaults to '.'.
3552²¹⁹³ Now, in case.py:
3553²¹⁹⁴ Original line:
3554²¹⁹⁵ path = os.path.join(self.test_directory, os.path.basename(filename))
3555²¹⁹⁶ Modified line:
3556²¹⁹⁷ path = os.path.join(self.test_directory, filename)
3557²¹⁹⁸ So the SEARCH/REPLACE block for case.py:
3558²¹⁹⁹ ### gabbi/case.py
3559²²⁰⁰ <<<<<<< SEARCH
3560²²⁰¹ path = os.path.join(self.test_directory, os.path.basename(filename))
3561²²⁰²)
3562²²⁰³ =====
3563²²⁰⁴

```

35642187 path = os.path.join(self.test_directory, filename)
35652188 >>>>>> REPLACE
35662189
35672190 Putting all these together, the final edits are:
35682191
35692192 1. Add import os in runner.py.
35702193
35712194 2. Modify run_suite to accept test_directory and pass it to
    test_suite_from_dict.
35722195
35732196 3. In the loop over input_files, compute test_directory as the
    directory of input_file and pass it to run_suite.
35742197
35752198 4. Modify case.py's path construction.
35762199
35772200 So the final answer would include these four SEARCH/REPLACE blocks.
35782201 </think>
35792202 ```python
35802203 ### gabbi/runner.py
35812204 <<<<<<< SEARCH
35822205 import argparse
35832206 from importlib import import_module
35842207 import sys
35852208 import unittest
35862209 =====
35872210 import argparse
35882211 from importlib import import_module
35892212 import os
35902213 import sys
35912214 import unittest
35922215 >>>>>> REPLACE
35932216 ```
35942217
35952218 ```python
35962219 ### gabbi/runner.py
35972220 <<<<<<< SEARCH
35982221 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
35992222 False,
36002223 failfast=False):
36012224 =====
36022225 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
36032226 False,
36042227 failfast=False, test_directory='.'):
36052228 >>>>>> REPLACE
36062229 ```
36072230
36082231 ```python
36092232 ### gabbi/runner.py
36102233 <<<<<<< SEARCH
36112234 test_suite = suitemaker.test_suite_from_dict(
36122235 loader, 'input', data, '.', host, port, None, None, prefix=
36132236 prefix,
36142237 handlers=handler_objects)
36152238 =====
36162239 test_suite = suitemaker.test_suite_from_dict(
36172240 loader, 'input', data, test_directory, host, port, None, None,
36182241 prefix=prefix, handlers=handler_objects)
36192242 >>>>>> REPLACE
36202243 ```
36212244
36222245 ```python
36232246 ### gabbi/runner.py
36242247 <<<<<<< SEARCH
36252248 else:

```

```

3618 2247         for input_file in input_files:
3619 2248             with open(input_file, 'r') as fh:
3620 2249                 success = run_suite(fh, handler_objects, host, port,
3621 2250                                 prefix, force_ssl, failfast)
3622 2251         =====
3623 2252         else:
3624 2253             for input_file in input_files:
3625 2254                 test_directory = os.path.dirname(input_file) or '.'
3626 2255                 with open(input_file, 'r') as fh:
3627 2256                     success = run_suite(fh, handler_objects, host, port,
3628 2257                                     prefix, force_ssl, failfast,
3629                                     test_directory=test_directory)
3630 2258
3631 2259 >>>>>> REPLACE
3632 2260
3633 2261     ```python
3634 2262     ### gabbi/case.py
3635 2263     <<<<<<< SEARCH
3636 2264         path = os.path.join(self.test_directory, os.path.basename(filename)
3637 2265                             )
3638 2266     =====
3639 2267         path = os.path.join(self.test_directory, filename)
3640 2268 >>>>>> REPLACE
3641 2269
3642 2270     ```
3643 2271
3644 2272 These changes:
3645 2273 1. Add 'os' import needed for path handling
3646 2274 2. Modify run_suite to accept a test_directory parameter
3647 2275 3. Use the input file's directory as the test_directory when processing
3648 2276     files
3649 2277 4. Update data file path resolution to preserve directory structure
3650 2278     from YAML relative paths
3651 2279 5. Remove basename handling that was stripping directory information
3652 2280     from data file paths
3653 2281
3654 2282 The net effect is that data files specified with '@filename' in YAML
3655 2283     tests will now be resolved relative to their YAML file's directory,
3656 2284     supporting both flat and nested directory structures for test data.
3657 2285     When tests are run from stdin (using '<'), the default directory
3658 2286     remains the current working directory as before.

```

Listing 20: Example of emergent parallel scaling (cdent_gabbi-191 in SWE-bench-extra).

H USE OF LARGE LANGUAGE MODELS

The initial draft of this paper was written entirely by the authors. A large language model (gpt-5) was used only to aid with polishing the language (e.g., grammar and clarity). All conceptual contributions, experimental designs, analyses, and conclusions are the work of the authors.