

000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 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-

054 implementations and the update of erroneous code snippets, as well as self-reflection and verification,
 055 all of which help scaffold the efficient adaptation of more capable and generalizable SWE-agents.
 056

057 Guided by this perspective, we introduce Kimi-Dev, an open-source code LLM for SWE tasks.
 058 Specifically, we first develop an Agentless training recipe, which includes mid-training, cold-start,
 059 reinforcement learning, and test-time self-play. This results in 60.4% accuracy on SWE-bench Verified,
 060 the SoTA performance among the workflow-based solutions. Building on this, we show that
 061 Agentless training induces skill priors: a minimal SFT cold-start from Kimi-Dev with 5k publicly-
 062 available trajectories enables efficient SWE-agent adaptation and reaches 48.6% pass@1 score, simi-
 063 lar to that of Claude 3.5 Sonnet (the 20241022 version, Anthropic (2024)). We demonstrate that
 064 these induced skills transfer from the non-agentic workflows to the agentic frameworks, and the
 065 self-reflection in long Chain-of-Thoughts baked through Agentless training further enable the agen-
 066 tic model to leverage more turns and succeed with a longer horizon. Finally, we also show that the
 067 skills from Agentless training generalize beyond SWE-bench Verified to broader benchmarks like
 068 SWE-bench-live (Zhang et al., 2025) and SWE-bench Multilingual (Yang et al., 2025c). Together,
 069 these results reframe the relationship between Agentless and agentic frameworks: not mutually ex-
 070 clusive, but as complementary stages in building transferable coding LLMs. This shift offers a prin-
 cipled view that training with structural skill priors could scaffold autonomous agentic interaction.
 071

072 The remainder of this paper is organized as follows. Section 2 reviews the background of the frame-
 073 work dichotomy and outlines the challenges of training SWE-Agents. Section 3 presents our Agent-
 074 less training recipe and the experimental results. Section 4 demonstrates how these Agentless-
 075 induced skill priors enable efficient SWE-Agent adaptation, and evaluates the skill transfer and
 076 generalization beyond SWE-bench Verified.
 077

2 BACKGROUND

079 In this section, we first review the two dominant frameworks for SWE tasks and their dichotomy in
 080 Section 2.1. We then summarize the progress and challenges of training SWE-Agents in Section 2.2.
 081 The background introduction sets the stage for reinterpreting Agentless training as skill priors for
 082 SWE-Agents, a central theme developed throughout the later sections.
 083

2.1 FRAMEWORK DICHOTOMY

085 Two paradigms currently dominate the solutions for automating software engineering tasks. Agent-
 086 less approaches decompose SWE tasks into modular workflows (Xia et al., 2024; Wei et al., 2025;
 087 Ma et al., 2025a;b; Xie et al., 2025). Typical workflows consist of bug localization, bug repair, and
 088 test generation. This design provides modularity and stability: each step could be optimized sepa-
 089 rately as a single-turn problem with verifiable rewards (Wei et al., 2025; He et al., 2025). However,
 090 such rigidity comes at the cost of flexibility. When encountering scenarios requiring multiple rounds
 091 of incremental updates, the Agentless approaches struggle to adapt.
 092

093 By contrast, SWE-agents adopt an end-to-end, multi-turn reasoning paradigm (Yang et al., 2024a;
 094 Wang et al., 2025a). Rather than following a fixed workflow, they iteratively plan, act, and reflect,
 095 resembling how human developers debug complex issues. This design enables greater adaptability,
 096 but introduces significant difficulties: trajectories often extend over tens or even hundreds of steps,
 097 context windows of the LLMs must span over the entire interaction history, and the model must
 098 handle exploration, reasoning, and tool use simultaneously.
 099

100 The dichotomy between fixed workflows (*e.g.*, Agentless) and agentic frameworks (*e.g.*, SWE-
 101 Agent) has shaped much of the community’s perspective. The two paradigms are often regarded
 102 as mutually exclusive: one trades off flexibility and performance ceiling for modularity and stabili-
 103 ty, whereas the other makes the reverse compromise. Our work challenges this dichotomy, as we
 104 demonstrate that Agentless training induces skill priors that make further SWE-agent training both
 105 more stable and more efficient.
 106

2.2 TRAINING SWE-AGENTS

107 Training SWE-agents relies on acquiring high-quality trajectories through interactions with exe-
 108 cutable environments. Constructing such large-scale environments and collecting reliable trajec-
 109

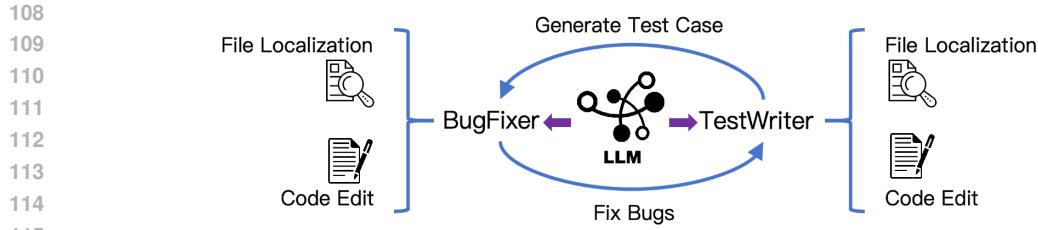


Figure 1: Agentless framework for Kimi-Dev: the duo of BugFixer and TestWriter.

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 $\sim 150B$ 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) $\sim 50B$ tokens in the form of Agentless derived from the

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

169 To activate the model’s long Chain-of-Thought (CoT) capability, we also construct a cold-start
 170 dataset with reasoning trajectories based on the SWE-Gym (Pan et al., 2024) and SWE-bench-extra
 171 (Badertdinov et al., 2024a) datasets, generated by the DeepSeek R1 model (Luo et al. (2025), the
 172 20250120 version). In this setup, R1 acts the roles of Bugfixer and Testwriter, producing out-
 173 puts such as file localization and code edits. Through supervised finetuning as a cold start with
 174 this dataset, we enable the model to acquire essential reasoning skills, including problem analysis,
 175 method sketching, self-refinement, and exploration of alternative solutions.

176 3.3 REINFORCEMENT LEARNING

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

182 For the RL algorithm, we adopt the policy optimization method proposed by Kimi k1.5 (Team et al.,
 183 2025), which has shown promising results on reasoning tasks in both math and coding. Unlike
 184 GRPO (Shao et al., 2024) that eliminates the critic by using average rewards for advantage estima-
 185 tion, Kimi k1.5 (Team et al., 2025) adopts a simpler policy gradient approach motivated by path
 186 consistency (Nachum et al., 2017). The specific RL objective is provided in Appendix C.1.

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

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

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

208 3.4 TEST-TIME SELF-PLAY

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

216
 217 Table 1: Performance comparison for models on SWE-bench Verified under Agentless-like frame-
 218 works. All the performances are obtained under the standard 40 patch, 40 test setting (Xia et al.,
 219 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

231
 232 from BugFixer, where the first run uses greedy decoding (temperature 0), and the remaining 39 use
 233 temperature 1 to ensure diversity. Similarly, 40 tests are generated independently from TestWriter.
 234 For the test patch candidates, to guarantee their validity, we first filter out those failing to raise a
 235 failure in the original repo without applying any BugFixer patch.

236 Denote the rest TestWriter patches as set \mathcal{T} , and the BugFixer patches as set \mathcal{B} . For each $b_i \in \mathcal{B}$
 237 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 ,
 238 and then with b_i applied. From the execution log for the first run, we get the count of the failed and
 239 the passed tests from t_j , denoted as $F(j)$ and $P(j)$. Comparing the execution logs for the two test
 240 suite runs, we get the count of the fail-to-pass and the pass-to-pass tests, denoted as $FP(i, j)$ and
 241 $PP(i, j)$, respectively. We then calculate the score for each b_i with

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

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

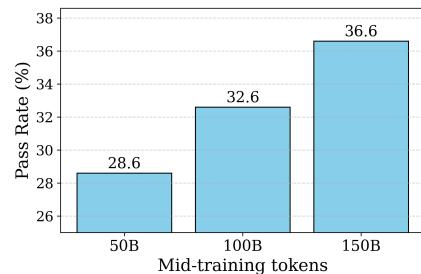
245 3.5 EXPERIMENTS

246 3.5.1 MAIN RESULTS

247 We evaluate our approach on SWE-bench Verified
 248 (Jimenez et al., 2023), which includes 500 GitHub issues
 249 guaranteed to be solved by human programmers. Table
 250 1 shows the overall results. Unlike SWE-RL (Wei et al.,
 251 2025), which relies on text-based similarity to the ground-
 252 truth patch, we use execution results as the reward signal,
 253 reducing shortcuts and improving solution quality. Unlike
 254 most Agentless approaches (Xia et al., 2024; Guo et al.,
 255 2025; He et al., 2025), which place a single-file reproduc-
 256 tion test at the root directory of the repository, we leverage
 257 the two-stage TestWriter to better capture repository con-
 258 text and align with human standard workflows (OpenAI,
 259 2025). Kimi-Dev model achieves the state-of-the-art per-
 260 formance among open-source models, resolving 60.4% of
 261 the issues in SWE-bench Verified.

262 3.5.2 MID-TRAINING

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



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

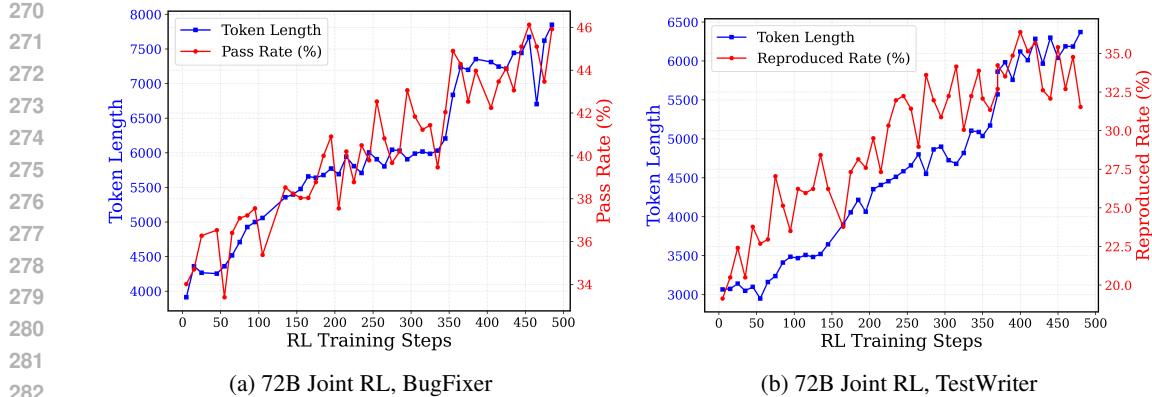


Figure 3: Joint code-edit RL experiments on the model after mid-training and cold-start. The pass rate for BugFixer and the reproduced rate for TestWriter are reported as pass@1 with temperature=1.0. The performance improves consistently as the output becomes increasingly longer.

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 (Badertdinov 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

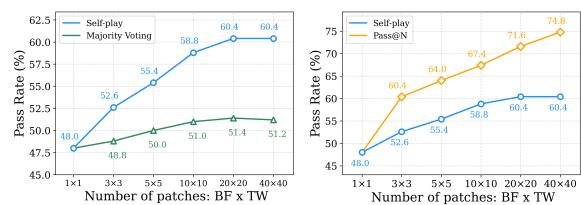


Figure 4: Test-time self-play on SWE-bench Verified. Performance improves with more generated patches and tests. Left: Execution-based self-play consistently surpasses BugFixer majority voting. Right: Self-play performances remain below pass@N where the ground-truth test patch is used, suggesting the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

324
 325 Table 2: Single-attempt performance of different models on SWE-bench Verified under end-to-end
 326 agentic frameworks, categorized by proprietary or open-weight models, and size over or under 100B.
 327 “Internal” denotes results achieved with their in-house agentic frameworks.

328 Model	329 System	330 #Params	331 Pass Rate (%)
<i>Proprietary</i>			
331 Gemini 2.5 Pro (Comanici et al., 2025)	332 Internal	333 -	334 60.3
332 OpenAI-o3 (OpenAI, 2025)	333 Internal	334 -	335 69.1
333 GPT-5 (OpenAI, 2025c)	334 Internal	335 -	336 74.9
334 Claude 3.5 Sonnet (241022) (Anthropic, 2024)	335 SWE-Agent	336 -	337 49.0
335 Claude 3.7 Sonnet (Anthropic, 2025a)	336 SWE-Agent	337 -	338 62.3
336 Claude 4.0 Sonnet (Anthropic, 2025b)	337 SWE-Agent	338 -	339 72.7
<i>Open Weight, $\geq 100B$</i>			
339 gpt-oss-120b (High) (OpenAI, 2025b)	340 Internal	341 120B	342 62.4
340 DeepSeek-v3.1 (Guo et al., 2025)	341 Internal	342 671B	343 66.0
341 Kimi-K2-0905 (Kimi et al., 2025)	342 SWE-Agent	343 1T	344 69.2
342 Qwen3-Coder (Yang et al., 2025a)	343 OpenHands	344 480B	345 69.6
<i>Open Weight, $< 100B$</i>			
345 Openhands-LM (Wang et al., 2025b)	346 OpenHands	347 32B	348 37.2
346 Skywork-SWE (Zeng et al., 2025)	347 OpenHands	348 32B	349 38.0
347 SWE-agent-LM (Yang et al., 2025b)	348 SWE-Agent	349 32B	350 40.2
348 DeepSWE (Luo et al., 2025)	349 OpenHands	350 32B	351 42.2
349 Devstral-Small-2507 (AI & AI, 2025)	350 OpenHands	351 24B	352 53.6
350 gpt-oss-20b (High) (OpenAI, 2025b)	351 Internal	352 20B	353 60.7
351 Kimi-Dev (SFTed)	352 SWE-Agent	353 72B	354 48.6

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

359 4 INITIALIZING SWE-AGENTS FROM AGENTLESS TRAINING

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

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

372 4.1 PERFORMANCE AFTER SWE-AGENT FINE-TUNING

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

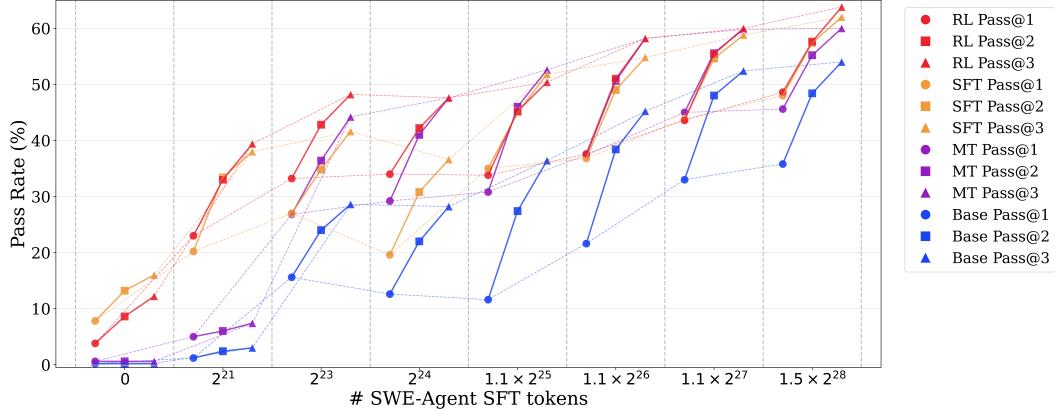


Figure 5: Comparing the quality of the raw Base, the Agentless mid-trained (MT), the Agentless mid-trained with reasoning-intensive cold-start (SFT), and the Kimi-Dev model after RL as the prior for SWE-Agent adaptation. The tokens of the SWE-Agent SFT trajectories are swept over different scales, and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

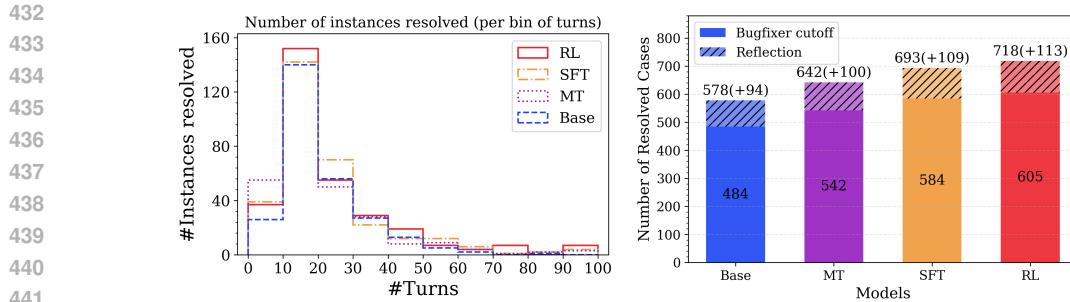


Figure 6: Left: Performance of the four priors under turn limits after SWE-Agent adaptation. Right: The characterization of the BugFixer and the reflection skills for each prior by counting the resolved cases of the 3 runs at Stage-3 cutoff moment, and comparing those with the final success cases.

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

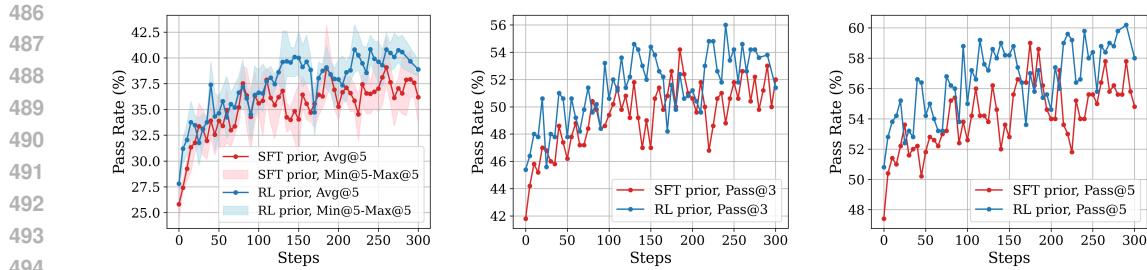


Figure 7: Comparison between the **SFT Prior** and the **RL Prior** when further applied with end-to-end SWE-Agent RL. Left: Pass@1 averaged from 5 runs. Middle: Pass@3. Right: Pass@5. The two priors are activated with the same 2^{21} SWE-Agent SFT tokens (the second column in Figure 5). After end-to-end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

540 ETHICS AND REPRODUCIBILITY STATEMENTS
541542 This work obeys the Code of Ethics required by the ICLR conference. The study does not involve
543 human subjects or animal experimentation. The personally identifiable information from raw data
544 is excluded for privacy consideration (see the mid-training data recipe detailed in Appendix A).545 For all of the experiments, we have covered the detailed setups in the appendices: mid-training for
546 Agentless in Appendix A, details of the used dockers in Appendix B, Agentless RL in Appendix C,
547 agent infrastructure in Appendix D, and case studies in Appendix F.
548549 REFERENCES
550551 Mistral AI and All Hands AI. Devstral-small-2507. [https://mistral.ai/news/](https://mistral.ai/news/devstral-2507)
552 devstral-2507, July 2025.553 Anthropic. Raising the bar on swe-bench verified with claude 3.5 sonnet. Online; AI model, Oct
554 2024. URL <https://www.anthropic.com/engineering/swe-bench-sonnet>.555 Anthropic. Claude 3.7 sonnet: Hybrid reasoning model. <https://www.anthropic.com/news/clause-3-7-sonnet>, February 2025a.556 Anthropic. Claude sonnet 4. <https://www.anthropic.com/news/clause-4>, May
557 2025b.
558559 Ibragim Badertdinov, Maria Trofimova, Yuri Anapolskiy, Sergey Abramov, Karina Zainullina,
560 Alexander Golubev, Sergey Polezhaev, Daria Litvintseva, Simon Karasik, Filipp Fisin, et al. Scaling
561 data collection for training software engineering agents. *Nebius blog*, 2024a.562 Ibragim Badertdinov, Maria Trofimova, Yury Anapolskiy, Sergey Abramov, Karina Zainullina,
563 Alexander Golubev, Sergey Polezhaev, Daria Litvintseva, Simon Karasik, Filipp Fisin, Sergey
564 Skvortsov, Maxim Nekrashevich, Anton Shevtsov, and Boris Yangel. Scaling data collection for
565 training software engineering agents. *Nebius blog*, 2024b.566 Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik,
567 Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. Swe-rebench:
568 An automated pipeline for task collection and decontaminated evaluation of software engineering
569 agents, 2025. URL <https://arxiv.org/abs/2505.20411>.570 Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhari-
571 wal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal,
572 Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M.
573 Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin,
574 Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford,
575 Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural*
576 *Information Processing Systems*, 33:1877–1901, 2020. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2005.14165)
577 2005.14165.578 Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, Omega, and
579 Kubernetes. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and*
580 *Implementation (OSDI '16)*, pp. 285–301, 2016.581 Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, Shu Liu, Eric Tang, Jiayi Pan, Xingyao
582 Wang, Akshay Malik, Graham Neubig, Kurosh Hakhamaneshi, Richard Liaw, Philipp Moritz,
583 Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-v0: Train real-world long-horizon
584 agents via reinforcement learning, 2025.585 Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu,
586 Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning
587 attention. *arXiv preprint arXiv:2506.13585*, 2025a.588 Mouxiang Chen, Binyuan Hui, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Jianling Sun, Junyang Lin,
589 and Zhongxin Liu. Parallel scaling law for language models. *arXiv preprint arXiv:2505.10475*,
590 2025b.

594 Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V
 595 Le, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of founda-
 596 tion model post-training. In *Forty-second International Conference on Machine Learning*, 2025.
 597 URL <https://openreview.net/forum?id=dYur3yabMj>.

598

599 Gheorghe Comanici, Eric Bieber, Mike Schaeckermann, Ice Pasupat, Noveen Sachdeva, Inderjit
 600 Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the
 601 frontier with advanced reasoning, multimodality, long context, and next generation agentic capa-
 602 bilities. *arXiv preprint arXiv:2507.06261*, 2025.

603

604 Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation
 605 of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

606

607 Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad
 608 Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd
 609 of models. *arXiv preprint arXiv:2407.21783*, 2024.

610

611 Xinran Gu, Kaifeng Lyu, Jiazheng Li, and Jingzhao Zhang. Data mixing can induce phase transitions
 612 in knowledge acquisition. *arXiv preprint arXiv:2505.18091*, 2025.

613

614 Arnav Gudibande, Eric Wallace, Charlie Victor Snell, Xinyang Geng, Hao Liu, Pieter Abbeel,
 615 Sergey Levine, and Dawn Song. The False Promise of Imitating Proprietary Language Mod-
 616 els. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=Kz3yckpCN5>.

617

618 Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,
 619 Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms
 620 via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

621

622 Zhenyu He, Qingping Yang, Wei Sheng, Xiaojian Zhong, Kechi Zhang, Chenxin An, Wenlei Shi,
 623 Tianle Cai, Di He, Jiaze Chen, Jingjing Xu, and Mingxuan Wang. Swe-swiss: A multi-task
 624 fine-tuning and rl recipe for high-performance issue resolution. Notion page, 2025.

625

626 Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2e-gym:
 627 Procedural environments and hybrid verifiers for scaling open-weights swe agents. *arXiv preprint
 628 arXiv:2504.07164*, 2025.

629

630 Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik
 631 Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint
 632 arXiv:2310.06770*, 2023.

633

634 Team Kimi, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen,
 635 Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv
 636 preprint arXiv:2507.20534*, 2025.

637

638 Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Ameen Patel, Qingyang Wu, Alpay Ariyak,
 639 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
 640 Agent from Scratch by Scaling RL. Notion page, 2025. Notion Blog.

641

642 Yingwei Ma, Rongyu Cao, Yongchang Cao, Yue Zhang, Jue Chen, Yibo Liu, Yuchen Liu, Binhua Li,
 643 Fei Huang, and Yongbin Li. Swe-gpt: A process-centric language model for automated software
 644 improvement. *Proc. ACM Softw. Eng.*, 2(ISTTA), June 2025a. doi: 10.1145/3728981. URL
 645 <https://doi.org/10.1145/3728981>.

646

647 Yingwei Ma, Qingping Yang, Rongyu Cao, Binhua Li, Fei Huang, and Yongbin Li. Alibaba ling-
 648 maagent: Improving automated issue resolution via comprehensive repository exploration. In
 649 *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engi-
 650 neering*, pp. 238–249, 2025b.

648 Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the
 649 gap between value and policy based reinforcement learning. In I. Guyon, U. Von
 650 Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Ad-*
 651 *vances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.,
 652 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/facf9f743b083008a894eee7baa16469-Paper.pdf.

654 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.

656

657 OpenAI. Introducing codex. <https://openai.com/index/introducing-codex/>, May
 658 2025.

659

660 OpenAI. Openai o3 and openai o4-mini system card. Technical report, OpenAI, Apr 2025. URL
 661 <https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf>.

662

663 OpenAI. Openai o3-mini system card. Technical report / system card, OpenAI, 2025a. URL
 664 <https://cdn.openai.com/o3-mini-system-card-feb10.pdf>. Description of
 665 o3-mini model, its safety evaluations and reasoning benchmarks.

666

667 OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025b. URL <https://arxiv.org/abs/2508.10925>.

668

669 OpenAI. Gpt-5 system card. OpenAI website, Aug 2025c. URL <https://cdn.openai.com/gpt-5-system-card.pdf>.

670

671 Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe
 672 Zhang. Training software engineering agents and verifiers with swe-gym. *arXiv preprint*
 673 *arXiv:2412.21139*, 2024.

674

675 Jiayi Pan, Xiuyu Li, Long Lian, Charlie Victor Snell, Yifei Zhou, Adam Yala, Trevor Darrell, Kurt
 676 Keutzer, and Alane Suhr. Learning adaptive parallel reasoning with language models. In *Second*
 677 *Conference on Language Modeling*, 2025. URL <https://openreview.net/forum?id=YgwQ7sXPXU>.

678

679 Qwen, ;, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan
 680 Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang,
 681 Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin
 682 Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,
 683 Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,
 684 Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2024.
 685 URL <https://arxiv.org/abs/2412.15115>.

686

687 ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang,
 688 Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, et al. Seed1. 5-thinking: Advancing superb rea-
 689 soning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*, 2025.

690

691 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,
 692 Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathemati-
 693 cal reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

694

695 Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun
 696 Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with
 697 llms. *arXiv preprint arXiv:2501.12599*, 2025.

698

699 Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan,
 700 Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng,
 701 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 soft-
 702 ware developers as generalist agents. In *The Thirteenth International Conference on Learning
 703 Representations*, 2025a. URL <https://openreview.net/forum?id=OJd3ayDDoF>.

702 Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan,
 703 Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng,
 704 Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert
 705 Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for AI soft-
 706 ware developers as generalist agents. In *The Thirteenth International Conference on Learning
 707 Representations*, 2025b. URL <https://openreview.net/forum?id=0Jd3ayDDoF>.

708 Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried,
 709 Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. Swe-rl: Advancing llm reasoning via rein-
 710 forcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025.

711 Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying
 712 llm-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024.

713 Chengxing Xie, Bowen Li, Chang Gao, He Du, Wai Lam, Difan Zou, and Kai Chen. SWE-fixer:
 714 Training open-source LLMs for effective and efficient GitHub issue resolution. In Wanxiang
 715 Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the
 716 Association for Computational Linguistics: ACL 2025*, pp. 1123–1139, Vienna, Austria, July
 717 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/
 718 v1/2025.findings-acl.62. URL [https://aclanthology.org/2025.findings-acl.
 719 62/](https://aclanthology.org/2025.findings-acl.62).

720 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu,
 721 Chang Gao, Chengan Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint
 722 arXiv:2505.09388*, 2025a.

723 John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan,
 724 and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering.
 725 *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024a.

726 John Yang, Kilian Lieret, Carlos E Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang,
 727 Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software
 728 engineering agents. *arXiv preprint arXiv:2504.21798*, 2025b.

729 John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang,
 730 Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software
 731 engineering agents, 2025c. URL <https://arxiv.org/abs/2504.21798>.

732 Zonghan Yang, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. ReAct Meets ActRe:
 733 Autonomous Annotation of Agent Trajectories for Contrastive Self-Training. In *First Con-
 734 ference on Language Modeling*, 2024b. URL <https://openreview.net/forum?id=0VLBwQGWpA>.

735 Liang Zeng, Yongcong Li, Yuzhen Xiao, Changshi Li, Chris Yuhao Liu, Rui Yan, Tianwen Wei,
 736 Juijie He, Xuchen Song, Yang Liu, et al. Skywork-swe: Unveiling data scaling laws for software
 737 engineering in llms. *arXiv preprint arXiv:2506.19290*, 2025.

738 Linghao Zhang, Shilin He, Chaoyun Zhang, Yu Kang, Bowen Li, Chengxing Xie, Junhao Wang,
 739 Maoquan Wang, Yufan Huang, Shengyu Fu, et al. Swe-bench goes live! *arXiv preprint
 740 arXiv:2505.23419*, 2025.

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

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 \sim 50B tokens for natural diff patches are obtained after applying these filtering rules.

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

```

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

```

Listing 1: Agentless prompt template: Localization for BugFixer.

```

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

```

Listing 2: Agentless prompt template: Localization for TestWriter.

```

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

```

```

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

```

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

```

893
894 85 We are currently solving the following issue within our repository.
895 86     Here is the issue text:
896 87     --- BEGIN ISSUE ---
897 88     {related issue / PR title content}
898 89     --- END ISSUE ---
899
900 91 Below are some code segments, each from a relevant test file. One or
901 92     more of these files may be added some new tests which can reproduce
902 93     the issue.
903 94
904 95     --- BEGIN FILE ---
905 96     '''
906 97     ##### {filename1}
907 98     {content of filename1}
908 99
909 100    ##### {filename2}
910 101    {content of filename2}
911 102    {...}
912 103
913 104    '''
914 105
915 106 Please first localize some possible locations in those test files
916 107     within the repo, and then generate *SEARCH/REPLACE* edit updates to
917 108     the **test** files in the repo, so that the erroneous scenario
918 109     described in the problem is reproduced.
919
920 110 Every *SEARCH/REPLACE* edit must use this format:
921 111 1. The file path
922

```

```

918 109 2. The start of search block: <<<<< SEARCH
919 110 3. A contiguous chunk of lines to search for in the existing source
920 111 code
921 112 4. The dividing line: =====
922 113 5. The lines to replace into the source code
923 114 6. The end of the replace block: >>>>> REPLACE
924 115
925 116 Here is an example:
926 117
927 118 '''python
928 119 ##### mathweb/flask/app.py
929 120 <<<<< SEARCH
930 121 from flask import Flask
931 122 =====
932 123 import math
933 124 from flask import Flask
934 125
935 126 def test__rules__std_L060_raised() -> None:
936 127     try:
937 128         sql = "SELECT IFNULL(NULL, 100),
938 129             NVL(NULL,100);"
939 130         result = lint(sql, rules=["L060"])
940 131         assert len(result) == 2
941 132     except:
942 133         print("Other issues")
943 134         return
944 135
945 136     try:
946 137         assert result[0]["description"] == "Use 'COALESCE' instead of 'IFNULL'."
947 138         assert result[1]["description"] == "Use 'COALESCE' instead of 'NVL'."
948 139         print("Issue resolved")
949 140     except AssertionError:
950 141         print("Issue reproduced")
951 142         return
952 143
953 144     return
954 145 >>>>> REPLACE
955 146
956 147 Please note that the *SEARCH/REPLACE* edit REQUIRES PROPER INDENTATION.
957 148     If you would like to add the line '           print(x)', you must
958 149     fully write that out, with all those spaces before the code!
960 150 Wrap the *SEARCH/REPLACE* edit in blocks '''python...'''.
961
962
963
964
965
966
967
968
969
970
971


```

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 \sim 20B tokens for PR commit packs are obtained after applying these filtering rules.

972 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 973 in the Agentless setting can be derived directly from the diff patch, since the set of files requiring 974 modification is explicitly indicated.
 975

976 **Synthetic reasoning data.** To improve reasoning quality, we perform a lightweight SFT of the 977 Qwen-2.5-72B-Instruct model on 2,000 R1 trajectories. The resulting model is then used to generate 978 large-scale rollouts for the localization stage of both BugFixer and TestWriter. We retain only the 979 rollouts that achieve exactly correct file localizations. This procedure yields approximately $\sim 10B$ 980 tokens of reasoning-intensive data dedicated to Agentless localization in the mid-training recipe.
 981

982 **Synthetic agentic interactions.** To strengthen agentic capabilities, we simulate agent-environment 983 interactions with a custom tool set designed to mimic file-system operations without execution. This 984 design is motivated by practical constraints: while repository snapshots from GitHub are available, 985 not all snapshots are equipped with an executable Docker environment. As a result, shell commands 986 are disabled. Instead, we introduce synthetic tools that allow the agent to view file contents and 987 perform keyword-based search for localization, which effectively reproduces the first stage of Agentless 988 but in an agentic manner. The specification of this tool set is covered in the system prompt, which is 989 then used to elicit agentic interaction rollouts from the Qwen-2.5-72B-Instruct model. The complete 990 system prompt is provided in Prompt 5. We apply a loss mask only to the system prompt, and enable 991 the model to simultaneously learn both actions and observations along the trajectory, inspired 992 by Yang et al. (2024b). This approach integrates both policy and world modeling into mid training.
 993

994 149 Your job is to look through the given GitHub problem description and
 995 150 Repository structure, and edit updates to the files in the repo to
 996 151 resolve the problem.
 997 152 The job is divided into two stages:
 998 153 + In Stage 1, you should localize the files the files that you would
 999 154 need to edit to fix the problem.
 1000 155 + In Stage 2, you should edit the updates to the repo.
 1001 156 Let's begin from Stage 1 to localize the bugs:
 1002 157
 1003 158 In Stage 1, besides reading the provided Repository structure, you can
 1004 159 use the following skills for exploration. The skills are to be
 1005 160 called in an environment wrapped by `<execute>` and `</execute>`, listed
 1006 161 in the form of python functions as below:
 1007 162
 1008 163 `open_file(path: str, is_all | None = False, line_number: int | None =`
 1009 164 `1, context_lines: int | None = 100) -> None:`
 1010 165 `Opens the file at the given path in the editor for exploration.`
 1011 166 `By default, only the first 100 lines of the file are displayed. To`
 1012 167 `open the entire file, set 'is_all' to 'True'.`
 1013 168 `The 'context_lines' parameter determines the maximum number of`
 1014 169 `lines to be displayed, with a cap of 100 lines. Use 'scroll_up'`
 1015 170 `and 'scroll_down' to view more content up or down.`
 1016 171 `If a 'line_number' is provided, the window will be moved to include`
 1017 172 `that line.`
 1018 173 `Note: When 'is_all' is set to 'True', the 'line_number' and '`
 1019 174 `context_lines' parameters will not take effect, as the entire`
 1020 175 `file will be opened and displayed without any line-specific`
 1021 176 `focus or context limitation.`
 1022 177 `Args:`
 1023 178 `path: str: The path to the file to open. the full path of the`
 1024 179 `filename should be provided.`
 1025 180 `is_all: bool | None = False: If set to 'True', the entire file will`
 1026 181 `be opened. Defaults to 'False'.`
 1027 182 `line_number: int | None = 1: The line number to move to. Defaults`
 1028 183 `to 1.`
 1029 184 `context_lines: int | None = 100: Only shows this number of lines in`
 1030 185 `the context window (usually from line 1), with line_number as`
 1031 186 `the center (if possible). Defaults to 100.`
 1032 187
 1033 188 `goto_line(line_number: int) -> None:`
 1034 189 `Moves the window to show the specified line number.`
 1035 190

```

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     scroll_down() -> None:
1036 181         Moves the window down by 100 lines.
1037 182     Args:
1038 183         None
1039 184     scroll_up() -> None:
1040 185         Moves the window up by 100 lines.
1041 186     Args:
1042 187         None
1043 188
1044 189     search_dir(search_term: str, dir_path: str | None) -> None:
1045 190         Searches for search_term in all files in dir. If dir is not
1046 191             provided, searches in the entire repository. Filename, fine-
1047 192                 grained line number, and the relative class or function it is
1048 193                     located in (if applied) will be shown for each found position.
1049 194     Args:
1050 195         search_term: str: The term to search for.
1051 196         dir_path: str: The path to the directory to search. Should be full
1052 197             path filename.
1053 198
1054 199     search_file(search_term: str, file_path: str | None = None) -> None:
1055 200         Searches for search_term in file. If file is not provided, searches
1056 201             in the current open file. Filename, fine-grained line number,
1057 202                 and the relative class or function it is located in (if applied)
1058 203                     will be shown for each found position.
1059 204     Args:
1060 205         search_term: str: The term to search for.
1061 206         file_path: str | None: The path to the file to search. Should be
1062 207             full path filename if provided.
1063 208
1064 209     find_file(file_name: str, dir_path: str | None) -> None:
1065 210         Finds all files with the given name in the specified directory. If
1066 211             dir is not provided, find in the entire repository.
1067 212     Args:
1068 213         file_name: str: The name of the file to find.
1069 214         dir_path: str: The path to the directory to search.
1070 215
1071 216     str_replace(path: str, old_str, new_str)
1072 217         old_str=[the old content to be replaced]
1073 218         new_str=[the new content after replacement]
1074 219         -> None:
1075 220             Replace the old content (old_str) in the file at the given path
1076 221                 with the new content (new_str). This is the skill that you will
1077 222                     be using to edit the updates.
1078 223     Args:
1079 224         path: str: The path to the file to be updated. The full path of the
1080 225             filename should be provided.
1081 226         old_str: str: The old content to be replaced. Note that this
1082 227             argument should be written in a new line starting with "old_str
1083 228                 =" , and the string content should not be quoted.
1084 229         new_str: str: The new content after replacement. Note that this
1085 230             argument should be written in a new line starting with "new_str
1086 231                 =" , and the string content should not be quoted.
1087 232
1088 233     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 'b\n' string in the 'filename.py' file.
1089 227
1090 228     insert(path: str, insert_line: int, new_str)
1091 229     new_str=[the new content to be inserted]
1092 230     -> None:
1093 231         Insert the new content (new_str) in the file at the given path.
1094 232         When you want to add an entirely new class/function to the file,
1095 233             it would be better to use this method.
1096 234         Args:
1097 235             path: str: The path to the file to be updated. The full path of the
1098 236                 filename should be provided.
1099 237             insert_line: int: The Line number below which the new content is to
1100 238                 be added. This Line number should be within the range of lines
1101 239                 of the file: [0, Lines_of_the_File]. Specifically, when
1102 240                     insert_line = 0, the added content starts from the top of the
1103 241                         file.
1104 242             new_str: str: The new content to be inserted. Note that this
1105 243                 argument should be written in a new line starting with "new_str"
1106 244                 ="", and the string content should not be quoted.
1107 245
1108 246         Example:
1109 247             Assuming a call is shown as follows:
1110 248             ...
1111 249             insert("test_filename.py", 5, new_str)
1112 250             new_str=    def test_add():
1113 251                 assert add(1, 2) == 3
1114 252             ...
1115 253             Then it will function as inserting the string '    def test_add():\n        assert add(1, 2) == 3' below the Line 5 of the 'test_filename.py' file.
1116 254
1117 255             stop() -> None:
1118 256                 Terminate the editing process.
1119 257             Args:
1120 258                 None
1121 259
1122 260             NOTE:
1123 261                 Responses should be concise.
1124 262                 When exploring, you should attempt fewer things at a time: Include ONLY
1125 263                     ONE <execute> per response, and use a SINGLE skill listed above
1126 264                     within the <execute> environment. DO NOT use other python functions,
1127 265                     as the environment does not support them.
1128 266                     You should first reason in the verbal form, then use a skill with <
1129 267                         execute> and </execute>.
1130 268                     You should avoid apologies and thanks in the responses.
1131 269
1132 270                     When you finish exploring and analyzing with the provided skills,
1133 271                         please return at most 3 files with the full path only. Each full
1134 272                         path should be placed in a single line, INSTEAD OF BROKEN WITH
1135 273                             MULTIPLE LINES.
1136 274                     The returned files should be separated by new lines ordered by most to
1137 275                         least important, wrapped with '```' and NOTHING ELSE.
1138 276                     An example for a full output:
1139 277                     ...
1140 278                     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.

1149 After completing the initial localization stage, the agent is guided into the code-editing phase
1150 through a follow-up instruction: “Now let’s move on to Stage 2 and edit the updates. Remember,
1151 you can still decide at any point whether a file actually requires modification.” We retain partial
1152 rollouts from Stage 1, provided that the localization results include at least one correct file.

1153 In Stage 2, we first simulate the agent’s interaction by allowing it to open incorrectly localized files,
1154 and we artificially inject agentic reasoning patterns such as “I realize that I do not need to modify
1155 this file” after inspecting the file content. This procedure is designed to strengthen the self-reflection
1156 ability of the agent by exposing it to false-positive contexts regarding the issue to be solved.

1157 Subsequently, we transcribe the ground-truth PR commit pack into trajectory form: each commit
1158 message is treated as the agent’s reasoning step, and each code update is represented as the cor-
1159 responding action, expressed through the “str_replace” or “insert” tools. These interactions are
1160 appended to the trajectory, followed by a terminating “stop” call. Due to storage constraints on
1161 repository snapshots, this trajectory simulation is applied to only a subset of PRs. Overall, this pro-
1162 cess contributes approximately $\sim 10B$ tokens of agentic interaction data to the mid-training recipe.
1163 Future directions for scaling this component in the data recipe include leveraging the idea of envi-
1164 ronment scaling (Yang et al., 2025c).

1165 **Training.** We perform mid-training using a standard next token prediction approach, initialized
1166 from the Qwen2.5-72B-Base (Qwen et al., 2024) model. We upsample the synthetic part of the data
1167 by a factor of 4 during mid-training, inspired by the practice in Grattafiori et al. (2024); Qwen et al.
1168 (2024); Gu et al. (2025). A global batch size of 256 with a maximum sequence length of 32K tokens
1169 is used, optimizing for long-context capabilities necessary for real-world software engineering tasks.
1170 The learning rate is set to 2e-5, with a cosine decay schedule and a minimum learning rate of 2e-6.
1171 The warm-up phase covers over approximately 3 billion tokens, followed by learning rate decay
1172 until approximately 150 billion tokens are processed. The training was conducted on 256 NVIDIA
1173 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/ https://huggingface.co/SWE-Gym/	2,356
SWE-bench-extra (Badertdinov et al. (2024a))	https://huggingface.co/datasets/nebius/ https://huggingface.co/SWE-bench-extra/	3,846
R2E-Gym-Lite (Jain et al. (2025))	https://huggingface.co/datasets/R2E-Gym/ https://huggingface.co/R2E-Gym-Lite	3,671

1188
 1189 **Docker environment construction.** To validate non-ground-truth patches generated by model roll-
 1190 outs and expand our dataset, we required executable Docker environments. We combined publicly
 1191 available datasets with custom-configured Docker environments (see Table. 3). Among them, SWE-
 1192 Gym and R2E-Gym-Lite open-source their dockers that we can directly use. For datasets lacking
 1193 Docker support (SWE-Bench-Extra), we implemented an automated configuration method:
 1194

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
 1199 passing (with `gt_patch`).

1200 Out of 6.38k SWE-bench-extra instances, 3,846 environments are successfully constructed and sub-
 1201 sequently used for cold-start and RL training.

1204 C MORE DETAILS OF RL TRAINING

1206 C.1 RL TRAINING OBJECTIVE

1208 For the RL algorithm, we adopt the policy optimization method proposed by Kimi k1.5 (Team et al.,
 1209 2025), which has shown promising results on reasoning tasks in both math and code. Unlike GRPO
 1210 (Shao et al., 2024), which eliminates the critic by using average rewards for advantage estimation,
 1211 Kimi k1.5 (Team et al., 2025) adopts a simpler policy gradient approach. Specifically, we optimize
 1212 the policy π_θ using the following objective:

$$1213 \mathcal{J}_\theta = \mathbb{E}_{q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_\theta(\cdot|q)} \left\{ \sum_{i=1}^G \left[\left(R_i(q, o_i) - \text{mean}(\{R_j\}_{j=1}^G) - \tau \log \frac{\pi_\theta(o_i|q)}{\pi_{\theta_{ref}}(o_i|q)} \right) \right] \right\} \quad (2)$$

1216 where q is a prompt sampled from the prompt data distribution $P(Q)$. $\{o_i\}_{i=1}^G$ are G responses
 1217 generated by the current policy π_θ for prompt q . $R_i(q, o_i)$ is the reward for response o_i . The
 1218 mean reward across the G responses is used as a baseline to center the reward (similar to advantage
 1219 estimation). The final term penalizes divergence from a reference policy $\pi_{\theta_{ref}}$, with weight τ . This
 1220 formulation avoids the need for a critic or value model by using group-wise relative rewards, and
 1221 supports stable updates through KL regularization.

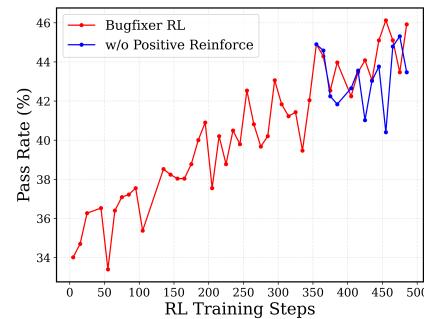
1223 C.2 PROMPT SET SELECTION

1224 In the main text, we introduce the adaptive prompt sele-
 1225 tion method for RL training. Specifically, we construct
 1226 an initial prompt set of 1,200 problems by selecting those
 1227 with $\text{pass}@16 > 0$ from SWE-Gym (Pan et al., 2024),
 1228 SWE-bench-extra (Badertdinov et al., 2025), and R2E-
 1229 gym (Jain et al., 2025). Then, every 100 training steps,
 1230 we expand the prompt set by adding 500 new problems.
 1231 These additional problems are randomly sampled from
 1232 the pool of problems for which the current model has
 1233 $\text{pass}@16 = 0$, thereby progressively increasing the dif-
 1234 ficulty.

1235 C.3 RL EXPERIMENT ABLATION

1236 Figure 9 shows the performance of the Qwen2.5-14B
 1237 model in RL experiments, where both the BugFixer and the TestWriter exhibit clear scaling law
 1238 behavior.

1239 Furthermore, Figure 8 illustrates the effect of our proposed positive example reinforcement. We
 1240 incorporated this improvement in the later stage of training and found that the RL of BugFixer



1241 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).

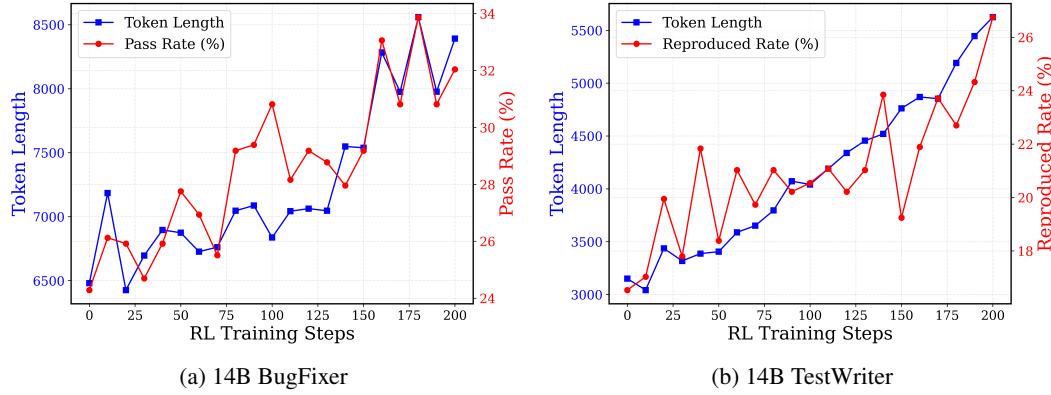


Figure 9: RL scaling experiments on Qwen2.5-14B model.

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:

```

...
Follow these steps to resolve the issue:
1. As a first step, it might be a good idea to find and read code
   relevant to the <pr_description>
2. Create a script to reproduce the error and execute it with 'python <
   filename.py>' using the bash tool, to confirm the error
3. Edit the source code of the repo to resolve the issue
4. Rerun your reproduce script and confirm that the error is fixed!
5. Think about edgecases and make sure your fix handles them as well
  
```

1296 284 ...

1298 Listing 6: The excerpt of the five-stage declaration in the SWE-Agent prompt.

1299

1300

1301 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.

1302

1303

1304

1305

1306

1307 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:

1311 285 You are a professional inspector that can analyze the provided agentic

1312 286 interaction trajectory.

1313 287

1314 288 The trajectory you are going to analyze is made by an agent that

1315 289 interacts with a computer to solve tasks. This agent has access to

1316 290 the following functions:

1317 291 ---- BEGIN FUNCTION #1: bash ----

1318 292 Description: Execute a bash command in the terminal.

1319 293 Parameters:

1320 294 (1) command (string, required): The bash command to execute. Can be

1321 295 empty to view additional logs when previous exit code is '-1'. Can

1322 296 be 'ctrl+c' to interrupt the currently running process.

1323 297 ---- END FUNCTION #1 ----

1324 298 ---- BEGIN FUNCTION #2: submit ----

1325 299 Description: Finish the interaction when the task is complete OR if the

1326 300 assistant cannot proceed further with the task.

1327 301 No parameters are required for this function.

1328 302 ---- END FUNCTION #2 ----

1329 303 ---- BEGIN FUNCTION #3: str_replace_editor ----

1330 304 Description: Custom editing tool for viewing, creating and editing

1331 305 files

1332 306 * State is persistent across command calls and discussions with the

1333 307 user

1334 308 * If 'path' is a file, 'view' displays the result of applying 'cat -n'.

1335 309 If 'path' is a directory, 'view' lists non-hidden files and

1336 310 directories up to 2 levels deep

1337 311 * The 'create' command cannot be used if the specified 'path' already

1338 312 exists as a file

1339 313 * If a 'command' generates a long output, it will be truncated and

1340 314 marked with '<response clipped>'

1341 315 * The 'undo_edit' command will revert the last edit made to the file at

1342 316 'path'

1343 317 Notes for using the 'str_replace' command:

1344 318 * The 'old_str' parameter should match EXACTLY one or more consecutive

1345 319 lines from the original file. Be mindful of whitespaces!

1346 320 * If the 'old_str' parameter is not unique in the file, the replacement

1347 321 will not be performed. Make sure to include enough context in '

1348 322 old_str' to make it unique

1349 323 * The 'new_str' parameter should contain the edited lines that should

1350 324 replace the 'old_str'

1351 325 Parameters:

```

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

```

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

1404 To provide a clearer understanding of the trajectory, we incorporate most of the tool descriptions and
 1405 instructions from the SWE-Agent system prompt into the annotation system prompt. The annotation
 1406 is conducted in a multi-round manner, leveraging the agent’s previous actions and observations, as
 1407 well as the stage classifications of earlier turns, to better exploit contextual information. At the
 1408 i -th round of annotation, the observation–action pair from turn i of the SWE-Agent trajectory is
 1409 appended as input, and the annotator is expected to output the corresponding stage classification.
 1410
 1411

1412 E.2 COMPARATIVE STUDY

1414 Based on the automatic stage annotation in the above section, we present a comparative study by
 1415 inspecting the performance on `sympy--sympy-20590` among the Kimi-Dev under Agentless, and
 1416 each of the Base, MT, SFT, and RL priors with SWE-Agent adaptation.
 1417

1418 The problem statement of `sympy--sympy-20590` is listed in Listing 8:

```
1419 345 Symbol instances have __dict__ since 1.7?  

  1420 346 In version 1.6.2 Symbol instances had no '__dict__' attribute  

  1421 347 '''python  

  1422 348 >>> sympy.Symbol('s').__dict__  

  1423 349  

  1424 350 AttributeError Traceback (most recent call last)  

  1425 351 <ipython-input-3-e2060d5eec73> in <module>  

  1426 352     >>> 1 sympy.Symbol('s').__dict__  

  1427 353  

  1428 354 AttributeError: 'Symbol' object has no attribute '__dict__'  

  1429 355 >>> sympy.Symbol('s').__slots__  

  1430 356 ('name',)  

  1431 357 ''  

  1432 358  

  1433 359 This changes in 1.7 where 'sympy.Symbol('s').__dict__' now exists (and  

  1434 360     returns an empty dict)  

  1435 I may misinterpret this, but given the purpose of '__slots__', I assume  

  1436     this is a bug, introduced because some parent class accidentally  

  1437     stopped defining '__slots__'.
```

1436 Listing 8: The problem statement of `sympy--sympy-20590`.
 1437

1438 It is observed that the main difficulty in resolving the issue lies in the realization of the “*some* parent
 1439 class” referenced in the problem. In fact, the hints text of this problem, which reflects the discussion
 1440 of the developers under the original issue, reveals a much more in-depth investigation into the issue
 1441 (Listing 9):
 1442

```
1443 361 It seems that Basic now inherits 'DefaultPrinting' which I guess doesn'  

  1444     t have slots. I'm not sure if it's a good idea to add '__slots__' to  

  1445     that class as it would then affect all subclasses.  

  1446 362  

  1447 363 ...  

  1448 364  

  1449 365 Using slots can break multiple inheritance but only if the slots are  

  1450     non-empty I guess. Maybe this means that any mixin should always  

  1451     declare empty slots or it won't work properly with subclasses that  

  1452     have slots...  

  1453 366  

  1454 367 I see that 'EvalfMixin' has '__slots__ = ()'.  

  1455 368 I guess we should add empty slots to DefaultPrinting then.
```

1455 Listing 9: The excerpted hints text of `sympy--sympy-20590`.
 1456

1457 According to the discussion, it is clear that the code repair would be to “add empty slots to Default-
 1458 Printing”, which naturally leads to the navigation towards the file related to the implementation of

1458 the printer (`sympy/core/_print_helpers.py`, which is also the file updated by the ground-
 1459 truth patch.) However, the `hints_text` information in the test set is *not* allowed to be used in the
 1460 problem-solving process, which challenges the reasoner or the agent to figure out “the parent class
 1461 that stopped defining `__slots__`” autonomously.

1462 We first examine Kimi-Dev under Agentless. None of the 40 runs succeeded in producing the
 1463 correct file localization. In most cases, the updates are made to `sympy/core/symbol.py`,
 1464 which is a plausible choice since the reported problem is triggered by `sympy.Symbol('s')`,
 1465 and `symbol.py` should contain the definition of the `Symbol` class. This line of reasoning, how-
 1466 ever, diverges from the correct solution. A small number of runs attempt alternative localizations,
 1467 such as `sympy/core/{basic.py, expr.py}`, but these are also incorrect.

1468 The limitation of the Agentless framework becomes more obvious when the focus is shifted to the
 1469 inspection over the SWE-Agent trajectories. For all of the Base, MT, SFT, and RL priors with
 1470 SWE-Agent adaptation, the resulting trajectories always contain a test-reproducing phase (marked
 1471 as Stage “2” by the annotator), with the generated script reads like:

```
1473 369 from sympy import Symbol
1474 370
1475 371 # Try to access __dict__ on a Symbol instance
1476 372 s = Symbol('s')
1477 373 print("Symbol.__slots__:", s.__slots__)
1478 374 try:
1479 375     print("Symbol.__dict__:", s.__dict__)
1480 376 except AttributeError as e:
1481 377     print("Error:", e)
```

1481 Listing 10: An example reproducing script in Stage 2 within the SWE-Agent trajectories in
 1482 `sympy-sympy-20590`.

1483

1484 It is trivial for an agent to generate a reproducing script of this form, as it directly transcribes the
 1485 case described in the problem statement. Executing the script provides an unambiguous signal of
 1486 whether the issue is resolved: the attribute `s.__dict__` should no longer exist, and the expected
 1487 `Error:` message should appear in the standard error output. Indeed, observing the initial comple-
 1488 tion of Stage 3 across all trajectories reveals that none of the agents succeeded on their first attempt.
 1489 However, the incomplete status is consistently exposed in Stage 4, when the script is rerun and
 1490 `s.__dict__` still persists. This progression highlights the advantage of the SWE-Agent framework
 1491 over the Agentless paradigm, as the agent is afforded additional opportunities for iterative refinement
 1492 rather than being limited to a single attempt.

1493 We continue to analyze the behavior of the agents adapted from the Base, MT, and SFT priors. Due
 1494 to the difficulty of the issue, it is non-trivial to correctly localize the error through their agentic
 1495 reasoning: the interactions show that the agents were switching among several file candidates, but
 1496 failed to reach the correct one. The only exception comes from the agent adapted from the RL prior,
 1497 which marks its unique success on this instance through accurate agentic reasoning. We attach the
 1498 intermediate process from the initial Stage-3 moment to the Stage-3 revisit (refining the initial code
 1499 repair) within the successful trajectory, tagged by the stage annotator:

```
1500 378 ...
1501 379
1502 380 OBSERVATION:
1503 381 Symbol.__slots__: ('name',)
1504 382 Symbol.__dict__: {}
1505 383
1506 384 ACTION:
1507 385 Hmm, it seems like the issue is still there. Let's check if there are
1508 386 any other classes in the hierarchy that might be missing '__slots__'
1509 387 `:
1510 388 <function=bash>
<parameter=command>python -c "from sympy import Symbol; s = Symbol('s')
1511 389 ; 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 394         __slots__'))  

1519 395
1520 396 SyntaxError: invalid syntax
1521 397
1522 398 ACTION:
1523 399 Let's try a different approach:
1524 400 <function=str_replace_editor>
1525 401 <parameter=command>create</parameter>
1526 402 <parameter=path>/testbed/check_slots.py</parameter>
1527 403 <parameter=file_text>
1528 404 from sympy import Symbol
1529 405
1530 406 s = Symbol('s')
1531 407 for cls in type(s).__mro__:
1532 408     print(f"{cls.__name__}:15} has __slots__: {hasattr(cls, '__slots__')}  

1533 409         }")
1534 410     if hasattr(cls, '__slots__'):
1535 411         print(f"{' '*16}__slots__ value: {cls.__slots__}")
1536 412 </parameter>
1537 413 </function>
1538 414 OBSERVATION:
1539 415 File created successfully at: /testbed/check_slots.py
1540 416
1541 417 ACTION:
1542 418 <function=bash>
1543 419 <parameter=command>python /testbed/check_slots.py</parameter>
1544 420 </function>
1545 421
1546 422 OBSERVATION:
1547 423 Symbol          has __slots__: True
1548 424             __slots__ value: ('name',)
1549 425 AtomicExpr      has __slots__: True
1550 426             __slots__ value: ()
1551 427 Atom            has __slots__: True
1552 428             __slots__ value: ()
1553 429 Expr             has __slots__: True
1554 430             __slots__ value: ()
1555 431 Boolean          has __slots__: True
1556 432             __slots__ value: ()
1557 433 Basic             has __slots__: True
1558 434             __slots__ value: ('_mhash', '_args', '_assumptions')
1559 435 Printable         has __slots__: False
1560 436 EvalfMixin       has __slots__: True
1561 437             __slots__ value: ()
1562 438 object            has __slots__: False
1563 439
1564 440 ACTION:
1565 441 Now I see the issue! 'Printable' doesn't have '__slots__' defined. Let'  

1566 442     s check the 'Printable' class:
1567 443 <function=str_replace_editor>
1568 444 <parameter=command>view</parameter>
1569 445 <parameter=path>/testbed/sympy/core/_print_helpers.py</parameter>
1570 446 <parameter=view_range>[1, 30]</parameter>
1571 447 </function>
1572 448
1573 449 OBSERVATION:

```

```

1566 450 Here's the result of running 'cat -n' on /testbed/sympy/core/
1567 451 _print_helpers.py:
1568 451
1569 452     1 """
1570 453     2 Base class to provide str and repr hooks that 'init_printing'
1571 454     3 can overwrite.
1572 455     4 This is exposed publicly in the 'printing.defaults' module,
1573 456     5 but cannot be defined there without causing circular imports.
1574 457     6 """
1575 458     7
1576 459     8 class Printable:
1577 460     9 """
1578 462     10     The default implementation of printing for SymPy classes.
1579 463     11
1580 463     12     This implements a hack that allows us to print elements of
1581 464     13     built-in
1582 465     14     Python containers in a readable way. Natively Python uses
1583 466     15     ``repr()`` even if ``str()`` was explicitly requested. Mix in this
1584 467     16     trait into
1585 468     17     a class to get proper default printing.
1586 469     18
1587 470     19
1588 470     20     # Note, we always use the default ordering (lex) in __str__
1589 471     21     and __repr__,
1590 472     22     # regardless of the global setting. See issue 5487.
1591 473     23     def __str__(self):
1592 474     24         from sympy.printing.str import sstr
1593 475     25         return sstr(self, order=None)
1594 476     26
1595 477     27     __repr__ = __str__
1596 478     28     def __repr_disabled(self):
1597 479     29         """
1598 480     30         No-op repr function used to disable jupyter display
1599 481     31     hooks.

1600 482 ACTION:
1601 483 We need to add '__slots__ = ()' to the 'Printable' class as well:
1602 484
1603 485 <function=str_replace_editor>
1604 486 <parameter=command>str_replace</parameter>
1605 488 <parameter=path>/testbed/sympy/core/_print_helpers.py</parameter>
1606 490 <parameter=old_str>
1607 491 class Printable:
1608 492     """
1609 493     The default implementation of printing for SymPy classes.
1610 494
1611 495     This implements a hack that allows us to print elements of built-in
1612 496     Python containers in a readable way. Natively Python uses ``repr()``
1613 497     ``
1614 498     even if ``str()`` was explicitly requested. Mix in this trait into
1615 499     a class to get proper default printing.
1616 500
1617 501     This also adds support for LaTeX printing in jupyter notebooks.
1618 502     """
1619 503 </parameter>
1620 504 <parameter=new_str>
1621 505 class Printable:
1622 506     """
1623 507     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 508 ``
1623 509 even if ``str()`` was explicitly requested. Mix in this trait into
1624 510 a class to get proper default printing.
1625 511
1626 512 This also adds support for LaTeX printing in jupyter notebooks.
1627 513 `"""
1628 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 'hist()'
1655 521 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 <function=bash>
1660 526 <parameter=command>find /testbed -type f -name "*.py" | grep -v "test_"
1661 527 " </parameter>
1662 528 </function>
1663 529 ACTION 1:
1664 530 Let's look for the histogram implementation since that's what we need
1665 531 to fix. The PR mentions that the bug is in the 'hist()' function
1666 532 when 'density=True' and 'range=' is specified.
1667 533 <function=bash>
1668 534 <parameter=command>grep -r "def hist" --include="*.py" /testbed | grep
1669 535 -v "test_" </parameter>
1670 536 </function>
1671 537 ACTION 2:
1672 538 The main histogram implementation appears to be in '/testbed/lib/
1673 539 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
ACTION 3:
1679 543 Let me look at the histogram implementation specifically:
1680 544
1681 546 <function=str_replace_editor>
1682 547 <parameter=command>view</parameter>
1683 548 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1684 549 <parameter=view_range>[6365, 6899]</parameter>
1685 550 </function>
1686 551
ACTION 4:
1687 553 Let me also look at the 'histogram_bin_edges' function since it's used
1688 554 in the code:
1689 555 <function=str_replace_editor>
1690 556 <parameter=command>view</parameter>
1691 557 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1692 558 <parameter=view_range>[40, 60]</parameter>
1693 559 </function>
1694 560
ACTION 5:
1695 561 2. Let's create a script to reproduce the error:
1696 562
1697 564 <function=str_replace_editor>
1698 565 <parameter=command>create</parameter>
1699 566 <parameter=path>/testbed/reproduce.py</parameter>
1700 567 ...
1701 568
ACTION 6:
1702 570 Let's run it to confirm the issue:
1703 571
1704 572 <function=bash>
1705 573 <parameter=command>python /testbed/reproduce.py</parameter>
1706 574 </function>
1707 576
ACTION 7:
1708 577 The issue is confirmed: when 'density=True' is specified, the 'range
1709 578 =(0, 1)' parameter is not respected. The bins should start at 0 and
1710 579 end at 1, but they don't.
1711 579
1712 580 3. Let's fix the issue. Looking at the code, the problem is in the 'hist'
1713 581 method and specifically around line 6689-6690:
1714 582
1715 583 '''python
1716 584 if density and not stacked:
1717 585     hist_kwargs = dict(density=density)
1718 586
1719 587 The issue is that when 'density=True', the 'range' parameter gets lost
1720 588 because we're not including it in the 'hist_kwargs'. Let's modify
1721 589 the code to fix this:
1722 590 <function=str_replace_editor>
1723 591 <parameter=command>str_replace</parameter>
1724 592 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1725 593 ...
ACTION 8:
1726 594 4. Let's run the reproduce script again to confirm the fix:
1727 595
1728 596 <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
ACTION 1:
1786 647 Hmm, the /testbed directory has a lot of files, but I need to focus on
1787 648 the code related to the hist function. The hist function is part of
1788 649 matplotlib.pyplot, so I should look for the pyplot module in the /
1789 650 testbed/lib/matplotlib directory.

1790 651 <function=str_replace_editor>
1791 652 <parameter=command>view</parameter>
1792 653 <parameter=path>/testbed/lib/matplotlib/pyplot.py</parameter>
1793 654 </function>

ACTION 2:
1795 655 Okay, I'll look at the hist function in pyplot.py.
1796 656
1797 657 <function=str_replace_editor>
1798 658 <parameter=command>view</parameter>
1799 659 <parameter=path>/testbed/lib/matplotlib/pyplot.py</parameter>
1800 660 <parameter=view_range>[2610, 2625]</parameter>
1801 661 </function>

ACTION 3:
1802 662 The hist function in pyplot.py simply calls the hist method of the
1803 663 current axes. So the actual implementation is in the Axes class.
1804 664
1805 665 I'll look for the Axes.hist method in the /testbed/lib/matplotlib/axes
1806 666 directory.

1807 667 <function=str_replace_editor>
1808 668 <parameter=command>view</parameter>
1809 669 <parameter=path>/testbed/lib/matplotlib/axes</parameter>
1810 670 </function>

ACTION 4:
1811 671 The Axes.hist method is likely in _axes.py, which is where most of the
1812 672 plotting methods are defined.
1813 673
1814 674 <function=str_replace_editor>
1815 675 <parameter=command>view</parameter>
1816 676 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1817 677 </function>

ACTION 5:
1818 678 I'll look at the hist method in _axes.py.
1819 679
1820 680 <function=str_replace_editor>
1821 681 <parameter=command>view</parameter>
1822 682 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1823 683 <parameter=view_range>[6365, 6400]</parameter>
1824 684 </function>

ACTION 6:
1825 685 I'll look at more of the hist method to see where the range is handled.
1826 686
1827 687 <function=str_replace_editor>
1828 688 <parameter=command>view</parameter>
1829 689 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1830 690 <parameter=view_range>[6400, 6450]</parameter>
1831 691 </function>

ACTION 7:
1832 692 I'll look at the code where the bins are calculated.
1833 693
1834 694
1835 695
701

```

```

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
ACTION 8:
1842 708 I'll look at the code where the bins are calculated.
1843 710
1844 711 <function=str_replace_editor>
1845 712 <parameter=command>view</parameter>
1846 713 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1847 714 <parameter=view_range>[6500, 6550]</parameter>
1848 715 </function>
1849 716
ACTION 9:
1850 718 I'll look at the code where the bins are calculated.
1851 719
1852 720 <function=str_replace_editor>
1853 721 <parameter=command>view</parameter>
1854 722 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1855 724 <parameter=view_range>[6550, 6600]</parameter>
1856 725 </function>
1857 726
ACTION 10:
1858 727 I'll look at the code where the bins are calculated.
1859 728
1860 730 <function=str_replace_editor>
1861 731 <parameter=command>view</parameter>
1862 732 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1863 733 <parameter=view_range>[6600, 6650]</parameter>
1864 734 </function>
1865 735
ACTION 11:
1866 736 I'll look at the code where the bins are calculated.
1867 737
1868 738 <function=str_replace_editor>
1869 739 <parameter=command>view</parameter>
1870 740 <parameter=path>/testbed/lib/matplotlib/axes/_axes.py</parameter>
1871 741 <parameter=view_range>[6650, 6700]</parameter>
1872 742 </function>
1873 743
ACTION 12:
1874 744 Ah, here's the key part. The code calculates the bins using np.
1875 745 histogram, and if the range is specified, it's passed to
1876 746 histogram_bin_edges. But when density is True, the code may be
1877 747 overriding the range.
1878 748
Wait, in the code:
1879 749
if not input_empty and len(x) > 1:
    if weights is not None:
        _w = np.concatenate(w)
    else:
        _w = None
1880 751
1881 752
1882 753
1883 754
1884 755
1885 756
1886 757
1887 758
1888 759
1889 760
1890 761
1891 762
1892 763
1893 764
1894 765
1895 766
1896 767
1897 768
1898 769
1899 770
1900 771
1901 772
1902 773
1903 774
1904 775
1905 776
1906 777
1907 778
1908 779
1909 780
1910 781
1911 782
1912 783
1913 784
1914 785
1915 786
1916 787
1917 788
1918 789
1919 790
1920 791
1921 792
1922 793
1923 794
1924 795
1925 796
1926 797
1927 798
1928 799
1929 800
1930 801
1931 802
1932 803
1933 804
1934 805
1935 806
1936 807
1937 808
1938 809
1939 810
1940 811
1941 812
1942 813
1943 814
1944 815
1945 816
1946 817
1947 818
1948 819
1949 820
1950 821
1951 822
1952 823
1953 824
1954 825
1955 826
1956 827
1957 828
1958 829
1959 830
1960 831
1961 832
1962 833
1963 834
1964 835
1965 836
1966 837
1967 838
1968 839
1969 840
1970 841
1971 842
1972 843
1973 844
1974 845
1975 846
1976 847
1977 848
1978 849
1979 850
1980 851
1981 852
1982 853
1983 854
1984 855
1985 856
1986 857
1987 858
1988 859
1989 860
1990 861
1991 862
1992 863
1993 864
1994 865
1995 866
1996 867
1997 868
1998 869
1999 870
2000 871
2001 872
2002 873
2003 874
2004 875
2005 876
2006 877
2007 878
2008 879
2009 880
2010 881
2011 882
2012 883
2013 884
2014 885
2015 886
2016 887
2017 888
2018 889
2019 890
2020 891
2021 892
2022 893
2023 894
2024 895
2025 896
2026 897
2027 898
2028 899
2029 900
2030 901
2031 902
2032 903
2033 904
2034 905
2035 906
2036 907
2037 908
2038 909
2039 910
2040 911
2041 912
2042 913
2043 914
2044 915
2045 916
2046 917
2047 918
2048 919
2049 920
2050 921
2051 922
2052 923
2053 924
2054 925
2055 926
2056 927
2057 928
2058 929
2059 930
2060 931
2061 932
2062 933
2063 934
2064 935
2065 936
2066 937
2067 938
2068 939
2069 940
2070 941
2071 942
2072 943
2073 944
2074 945
2075 946
2076 947
2077 948
2078 949
2079 950
2080 951
2081 952
2082 953
2083 954
2084 955
2085 956
2086 957
2087 958
2088 959
2089 960
2090 961
2091 962
2092 963
2093 964
2094 965
2095 966
2096 967
2097 968
2098 969
2099 970
2100 971
2101 972
2102 973
2103 974
2104 975
2105 976
2106 977
2107 978
2108 979
2109 980
2110 981
2111 982
2112 983
2113 984
2114 985
2115 986
2116 987
2117 988
2118 989
2119 990
2120 991
2121 992
2122 993
2123 994
2124 995
2125 996
2126 997
2127 998
2128 999
2129 1000
2130 1001
2131 1002
2132 1003
2133 1004
2134 1005
2135 1006
2136 1007
2137 1008
2138 1009
2139 1010
2140 1011
2141 1012
2142 1013
2143 1014
2144 1015
2145 1016
2146 1017
2147 1018
2148 1019
2149 1020
2150 1021
2151 1022
2152 1023
2153 1024
2154 1025
2155 1026
2156 1027
2157 1028
2158 1029
2159 1030
2160 1031
2161 1032
2162 1033
2163 1034
2164 1035
2165 1036
2166 1037
2167 1038
2168 1039
2169 1040
2170 1041
2171 1042
2172 1043
2173 1044
2174 1045
2175 1046
2176 1047
2177 1048
2178 1049
2179 1050
2180 1051
2181 1052
2182 1053
2183 1054
2184 1055
2185 1056
2186 1057
2187 1058
2188 1059
2189 1060
2190 1061
2191 1062
2192 1063
2193 1064
2194 1065
2195 1066
2196 1067
2197 1068
2198 1069
2199 1070
2200 1071
2201 1072
2202 1073
2203 1074
2204 1075
2205 1076
2206 1077
2207 1078
2208 1079
2209 1080
2210 1081
2211 1082
2212 1083
2213 1084
2214 1085
2215 1086
2216 1087
2217 1088
2218 1089
2219 1090
2220 1091
2221 1092
2222 1093
2223 1094
2224 1095
2225 1096
2226 1097
2227 1098
2228 1099
2229 1100
2230 1101
2231 1102
2232 1103
2233 1104
2234 1105
2235 1106
2236 1107
2237 1108
2238 1109
2239 1110
2240 1111
2241 1112
2242 1113
2243 1114
2244 1115
2245 1116
2246 1117
2247 1118
2248 1119
2249 1120
2250 1121
2251 1122
2252 1123
2253 1124
2254 1125
2255 1126
2256 1127
2257 1128
2258 1129
2259 1130
2260 1131
2261 1132
2262 1133
2263 1134
2264 1135
2265 1136
2266 1137
2267 1138
2268 1139
2269 1140
2270 1141
2271 1142
2272 1143
2273 1144
2274 1145
2275 1146
2276 1147
2277 1148
2278 1149
2279 1150
2280 1151
2281 1152
2282 1153
2283 1154
2284 1155
2285 1156
2286 1157
2287 1158
2288 1159
2289 1160
2290 1161
2291 1162
2292 1163
2293 1164
2294 1165
2295 1166
2296 1167
2297 1168
2298 1169
2299 1170
2300 1171
2301 1172
2302 1173
2303 1174
2304 1175
2305 1176
2306 1177
2307 1178
2308 1179
2309 1180
2310 1181
2311 1182
2312 1183
2313 1184
2314 1185
2315 1186
2316 1187
2317 1188
2318 1189
2319 1190
2320 1191
2321 1192
2322 1193
2323 1194
2324 1195
2325 1196
2326 1197
2327 1198
2328 1199
2329 1200
2330 1201
2331 1202
2332 1203
2333 1204
2334 1205
2335 1206
2336 1207
2337 1208
2338 1209
2339 1210
2340 1211
2341 1212
2342 1213
2343 1214
2344 1215
2345 1216
2346 1217
2347 1218
2348 1219
2349 1220
2350 1221
2351 1222
2352 1223
2353 1224
2354 1225
2355 1226
2356 1227
2357 1228
2358 1229
2359 1230
2360 1231
2361 1232
2362 1233
2363 1234
2364 1235
2365 1236
2366 1237
2367 1238
2368 1239
2369 1240
2370 1241
2371 1242
2372 1243
2373 1244
2374 1245
2375 1246
2376 1247
2377 1248
2378 1249
2379 1250
2380 1251
2381 1252
2382 1253
2383 1254
2384 1255
2385 1256
2386 1257
2387 1258
2388 1259
2389 1260
2390 1261
2391 1262
2392 1263
2393 1264
2394 1265
2395 1266
2396 1267
2397 1268
2398 1269
2399 1270
2400 1271
2401 1272
2402 1273
2403 1274
2404 1275
2405 1276
2406 1277
2407 1278
2408 1279
2409 1280
2410 1281
2411 1282
2412 1283
2413 1284
2414 1285
2415 1286
2416 1287
2417 1288
2418 1289
2419 1290
2420 1291
2421 1292
2422 1293
2423 1294
2424 1295
2425 1296
2426 1297
2427 1298
2428 1299
2429 1300
2430 1301
2431 1302
2432 1303
2433 1304
2434 1305
2435 1306
2436 1307
2437 1308
2438 1309
2439 1310
2440 1311
2441 1312
2442 1313
2443 1314
2444 1315
2445 1316
2446 1317
2447 1318
2448 1319
2449 1320
2450 1321
2451 1322
2452 1323
2453 1324
2454 1325
2455 1326
2456 1327
2457 1328
2458 1329
2459 1330
2460 1331
2461 1332
2462 1333
2463 1334
2464 1335
2465 1336
2466 1337
2467 1338
2468 1339
2469 1340
2470 1341
2471 1342
2472 1343
2473 1344
2474 1345
2475 1346
2476 1347
2477 1348
2478 1349
2479 1350
2480 1351
2481 1352
2482 1353
2483 1354
2484 1355
2485 1356
2486 1357
2487 1358
2488 1359
2489 1360
2490 1361
2491 1362
2492 1363
2493 1364
2494 1365
2495 1366
2496 1367
2497 1368
2498 1369
2499 1370
2500 1371
2501 1372
2502 1373
2503 1374
2504 1375
2505 1376
2506 1377
2507 1378
2508 1379
2509 1380
2510 1381
2511 1382
2512 1383
2513 1384
2514 1385
2515 1386
2516 1387
2517 1388
2518 1389
2519 1390
2520 1391
2521 1392
2522 1393
2523 1394
2524 1395
2525 1396
2526 1397
2527 1398
2528 1399
2529 1400
2530 1401
2531 1402
2532 1403
2533 1404
2534 1405
2535 1406
2536 1407
2537 1408
2538 1409
2539 1410
2540 1411
2541 1412
2542 1413
2543 1414
2544 1415
2545 1416
2546 1417
2547 1418
2548 1419
2549 1420
2550 1421
2551 1422
2552 1423
2553 1424
2554 1425
2555 1426
2556 1427
2557 1428
2558 1429
2559 1430
2560 1431
2561 1432
2562 1433
2563 1434
2564 1435
2565 1436
2566 1437
2567 1438
2568 1439
2569 1440
2570 1441
2571 1442
2572 1443
2573 1444
2574 1445
2575 1446
2576 1447
2577 1448
2578 1449
2579 1450
2580 1451
2581 1452
2582 1453
2583 1454
2584 1455
2585 1456
2586 1457
2587 1458
2588 1459
2589 1460
2590 1461
2591 1462
2592 1463
2593 1464
2594 1465
2595 1466
2596 1467
2597 1468
2598 1469
2599 1470
2600 1471
2601 1472
2602 1473
2603 1474
2604 1475
2605 1476
2606 1477
2607 1478
2608 1479
2609 1480
2610 1481
2611 1482
2612 1483
2613 1484
2614 1485
2615 1486
2616 1487
2617 1488
2618 1489
2619 1490
2620 1491
2621 1492
2622 1493
2623 1494
2624 1495
2625 1496
2626 1497
2627 1498
2628 1499
2629 1500
2630 1501
2631 1502
2632 1503
2633 1504
2634 1505
2635 1506
2636 1507
2637 1508
2638 1509
2639 1510
2640 1511
2641 1512
2642 1513
2643 1514
2644 1515
2645 1516
2646 1517
2647 1518
2648 1519
2649 1520
2650 1521
2651 1522
2652 1523
2653 1524
2654 1525
2655 1526
2656 1527
2657 1528
2658 1529
2659 1530
2660 1531
2661 1532
2662 1533
2663 1534
2664 1535
2665 1536
2666 1537
2667 1538
2668 1539
2669 1540
2670 1541
2671 1542
2672 1543
2673 1544
2674 1545
2675 1546
2676 1547
2677 1548
2678 1549
2679 1550
2680 1551
2681 1552
2682 1553
2683 1554
2684 1555
2685 1556
2686 1557
2687 1558
2688 1559
2689 1560
2690 1561
2691 1562
2692 1563
2693 1564
2694 1565
2695 1566
2696 1567
2697 1568
2698 1569
2699 1570
2700 1571
2701 1572
2702 1573
2703 1574
2704 1575
2705 1576
2706 1577
2707 1578
2708 1579
2709 1580
2710 1581
2711 1582
2712 1583
2713 1584
2714 1585
2715 1586
2716 1587
2717 1588
2718 1589
2719 1590
2720 1591
2721 1592
2722 1593
2723 1594
2724 1595
2725 1596
2726 1597
2727 1598
2728 1599
2729 1600
2730 1601
2731 1602
2732 1603
2733 1604
2734 1605
2735 1606
2736 1607
2737 1608
2738 1609
2739 1610
2740 1611
2741 1612
2742 1613
2743 1614
2744 1615
2745 1616
2746 1617
2747 1618
2748 1619
2749 1620
2750 1621
2751 1622
2752 1623
2753 1624
2754 1625
2755 1626
2756 1627
2757 1628
2758 1629
2759 1630
2760 1631
2761 1632
2762 1633
2763 1634
2764 1635
2765 1636
2766 1637
2767 1638
2768 1639
2769 1640
2770 1641
2771 1642
2772 1643
2773 1644
2774 1645
2775 1646
2776 1647
2777 1648
2778 1649
2779 1650
2780 1651
2781 1652
2782 1653
2783 1654
2784 1655
2785 1656
2786 1657
2787 1658
2788 1659
2789 1660
2790 1661
2791 1662
2792 1663
2793 1664
2794 1665
2795 1666
2796 1667
2797 1668
2798 1669
2799 1670
2800 1671
2801 1672
2802 1673
2803 1674
2804 1675
2805 1676
2806 1677
2807 1678
2808 1679
2809 1680
2810 1681
2811 1682
2812 1683
2813 1684
2814 1685
2815 1686
2816 1687
2817 1688
2818 1689
2819 1690
2820 1691
2821 1692
2822 1693
2823 1694
2824 1695
2825 1696
2826 1697
2827 1698
2828 1699
2829 1700
2830 1701
2831 1702
2832 1703
2833 1704
2834 1705
2835 1706
2836 1707
2837 1708
2838 1709
2839 1710
2840 1711
2841 1712
2842 1713
2843 1714
2844 1715
2845 1716
2846 1717
2847 1718
2848 1719
2849 1720
2850 1721
2851 1722
2852 1723
2853 1724
2854 1725
2855 1726
2856 1727
2857 1728
2858 1729
2859 1730
2860 1731
2861 1732
2862 1733
2863 1734
2864 1735
2865 1736
2866 1737
2867 1738
2868 1739
2869 1740
2870 1741
2871 1742
2872 1743
2873 1744
2874 1745
2875 1746
2876 1747
2877 1748
2878 1749
2879 1750
2880 1751
2881 1752
2882 1753
2883 1754
2884 1755
2885 1756
2886 1757
2887 1758
2888 1759
2889 1760
2890 1761
2891 1762
2892 1763
2893 1764
2894 1765
2895 1766
2896 1767
2897 1768
2898 1769
2899 1770
2900 1771
2901 1772
2902 1773
2903 1774
2904 1775
2905 1776
2906 1777
2907 1778
2908 1779
2909 1780
2910 1781
2911 1782
2912 1783
2913 1784
2914 1785
2915 1786
2916 1787
2917 1788
2918 1789
2919 1790
2920 1791
2921 1792
2922 1793
2923 1794
2924 1795
2925 1796
2926 1797
2927 1798
2928 1799
2929 1800
2930 1801
2931 1802
2932 1803
2933 1804
2934 1805
2935 1806
2936 1807
2937 1808
2938 1809
2939 1810
2940 1811
2941 1812
2942 1813
2943 1814
294
```

```

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_kw_args['range'] = bin_range
1920 793
1921 794 density = bool(density) or bool(normed)
1922 795 if density and not stacked:
1923 796     hist_kw_args = 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_kw_args)
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.
1953 826

```

```

1944 822 This suggests that when density=True, the code is not using the
1945 823 specified range for the bins.
1946 823
1947 824 But why?
1948 825
1949 826 Let me look at the code again.
1950 827
1951 828 When the code calculates the bins:
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
1967 844 Then, in the loop:
1968 845
1969 846     for i in range(nx):
1970 847         m, bins = np.histogram(x[i], bins, weights=w[i], **hist_kwargs)
1971 848
1972 849 But wait, if bins is None, then np.histogram will use the range. But if
1973 850 bins is a string (like 'auto'), then np.histogram will calculate
1974 851 the bins based on the data and the range.
1975 852
1976 853 Wait, in the user's code, bins is 'auto' and range is (0, 1). When
1977 854 density=True, the code should calculate the bins based on the range
1978 855 (0, 1).
1979 856
1980 857 But in the code, when len(x) is 1, the code sets hist_kwargs['range'] =
1981 858 bin_range. Then, in the loop, the code calls np.histogram(x[i],
1982 859 bins, weights=w[i], **hist_kwargs).
1983 860
1984 861 But if bins is a string (like 'auto'), then np.histogram will calculate
1985 862 the bins based on the data and the range. So the range should be
1986 863 respected.
1987 864
1988 865 But in the user's code, the bins are not from 0 to 1. This suggests
1989 866 that the code is not passing the range to np.histogram when density=
1990 867 True.
1991 868
1992 869 But why?
1993 870
1994 871 This is puzzling. Let me try to reproduce the error.
1995 872
1996 873 I'll create a script in the /testbed directory.
1997 874
1998 875 <function=str_replace_editor>
1999 876 <parameter=command>create</parameter>
2000 877 <parameter=path>/testbed/reproduce_error.py</parameter>
2001 878 ...
2002 879
2003 880 ACTION 13:
2004 881 I'll run the script.
2005 882
2006 883 <function=bash>
2007 884 <parameter=command>python /testbed/reproduce_error.py</parameter>
2008 885 </function>

```

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
 2005 882 Let me look at the code again.
 2006 883 When len(x) is 1, the code sets hist_kwarg['range'] = bin_range. Then,
 2007 884 in the loop:
 2008 885
 2009 886 for i in range(nx):
 2010 887 m, bins = np.histogram(x[i], bins, weights=w[i], **hist_kwarg)
 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
 2018 895 But why?
 2019 896 This is confusing. Let me look at the code for np.histogram.
 2020 897
 2021 898 The np.histogram function has a range parameter. If range is specified,
 2022 899 the bin edges are calculated based on that range. If the data is
 2023 900 outside the range, it is ignored.
 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
 2031 908 But why?
 2032 909 This is puzzling. Let me try to modify the code in _axes.py.
 2033 910 I'll look for the line where hist_kwarg 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_kwarg = dict(density=density)
 2040 917
 2041 918 This line is overwriting hist_kwarg, which may include the range.
 2042 919 Ah, here's the problem. If density is True and stacked is False, the
 2043 920 code sets hist_kwarg to dict(density=density), which overwrites any
 2044 921 previous hist_kwarg, including the range.
 2045 922
 2046 923 This is the root of the problem. The code is overwriting hist_kwarg,
 2047 924 which includes the range, when density is True and stacked is False.
 2048 925 So, in the code:
 2049 926
 2050 927 density = bool(density) or bool(normed)
 2051 928 if density and not stacked:
 2052 929 hist_kwarg = 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 927
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 929
2058 930 So, the code should be:
2059 931
2060 932 density = bool(density) or bool(normed)
2061 933 if density and not stacked:
2062 934     hist_kwargs.setdefault('density', density)
2063 935
2064 936 But wait, hist_kwargs may already have the range. So, the code should
2065 937 not overwrite hist_kwargs, but add to it.
2066 938
2067 939 The code is:
2068 940
2069 941 density = bool(density) or bool(normed)
2070 942 if density and not stacked:
2071 943     hist_kwargs = dict(density=density)
2072 944
2073 945 This line is overwriting hist_kwargs, which may include the range. So
2074 946 the range is lost.
2075 947
2076 948 The solution is to modify this code to add to hist_kwargs, not
2077 949 overwrite it.
2078 950
2079 951 So, the code should be:
2080 952
2081 953 density = bool(density) or bool(normed)
2082 954 if density and not stacked:
2083 955     hist_kwargs.setdefault('density', density)
2084 956
2085 957 But wait, hist_kwargs may not exist. Let me look at the code.
2086 958
2087 959 The code for hist_kwargs is:
2088 960
2089 961 hist_kwargs = dict()
2090 962
2091 963 # if the bin_range is not given, compute without nan numpy
2092 964 # does not do this for us when guessing the range (but will
2093 965 # happily ignore nans when computing the histogram).
2094 966 if bin_range is None:
2095 967     xmin = np.inf
2096 968     xmax = -np.inf
2097 969     for xi in x:
2098 970         if len(xi):
2099 971             xmin = min(xmin, np.nanmin(xi))
2100 972             xmax = max(xmax, np.nanmax(xi))
2101 973
2102 974     # make sure we have seen at least one non-nan and finite
2103 975     # value before we reset the bin range
2104 976     if not np.isnan([xmin, xmax]).any() and not (xmin > xmax):
2105 977         bin_range = (xmin, xmax)
2106 978
2107 979 # If bins are not specified either explicitly or via range,
2108 980 # we need to figure out the range required for all datasets,
2109 981 # and supply that to np.histogram.
2110 982 if not input_empty and len(x) > 1:
2111 983     if weights is not None:
2112 984         _w = np.concatenate(w)
2113 985     else:
2114 986         _w = None
2115 987
2116 988 bins = histogram_bin_edges(np.concatenate(x),
2117 989                         bins, bin_range, _w)

```

```

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

2165 Listing 13: The interaction made by the RL prior with 2^{21} SWE-Agent SFT tokens and end-to-end
 2166 SWE-Agent RL on `matplotlib_matplotlib-13989`.
 2167

2168
 2169 It can be seen that the two types of patterns are significantly different from each other:
 2170

- 2172 • The patterns of the heavy SFT model appear to be more regulated, with explicitly labeled
 2173 steps 1 to 5 along the intermediate reasoning process. The reasoning content is generally
 2174 short as well. This reflects the behavior of the model shaped by the heavy SFT data from
 2175 the proprietary models.
- 2177 • The patterns of the light SFT + end-to-end RL model, in contrast, are much more under-
 2178 regulated. More turns of interaction are spent at repo exploration, and a significantly larger
 2179 amount of tokens are generated at the code edit steps before the action is made. This is the
 2180 reflection of behavior shaped by end-to-end RL (and less effect from expert data), as the
 2181 sign of reasoning-intensive skill priors is retained in the agentic scenario.

2183 E.4 GENERALIZATION TO OTHER TASKS

2185 In this section, we study whether the skills incentivized by Agentless training and SWE-Agent adap-
 2186 tion through SWE-Smith trajectories could transfer to other SWE tasks. We use SWE-bench-live
 2187 and SWE-bench Multilingual as our testbeds.

2188 SWE-bench-live (Zhang et al., 2025) is a benchmark for real-world issue resolution, evaluating AI
 2189 systems on software engineering tasks. Using an automated curation pipeline, it is updated monthly
 2190 to ensure fresh tasks and rigorous, contamination-free evaluation. For our experiments, we selected
 2191 the default set of 300 tasks, with data collected between October 2024 and March 2025. Compared
 2192 to SWE-bench Verified, SWE-bench-live exhibits a higher degree of distributional shift.

2193 SWE-bench Multilingual (Yang et al., 2025c) introduces 300 curated tasks from 42 GitHub repos-
 2194 ositories across 9 programming languages, including Rust, Java, PHP, Ruby, JavaScript/TypeScript,
 2195 Go, and C/C++, covering domains such as web frameworks, data tools, core utilities, and libraries.
 2196 Compared to SWE-bench Verified, which focuses exclusively on Python, SWE-bench Multilingual
 2197 exhibits greater linguistic and domain diversity, posing additional challenges in cross-language gen-
 2198 eralization and transferability of software engineering capabilities.

2199 Similar to previous experiments, we evaluated four model stages as the priors: the original Qwen2.5-
 2200 72B (Base), the mid-trained model (MT), the model activated with reasoning data through super-
 2201 vised finetuning (SFT), and the model after RL training (RL). We still use the open-source SWE-
 2202 smith trajectories to activate the agentic capabilities of each prior.

2203 Figures 10 and 11 show the performance of the four priors on SWE-bench-Live and SWE-bench
 2204 Multilingual under varied amounts of agentic trajectories for adaptation (2^{21} as one-step gradient
 2205 descent, 2^{23} , 1.1×2^{27} , and 1.5×2^{28} as 100, 2,000, and 5,016 training trajectories). Each SWE-
 2206 Agent adaptation experiment is conducted through lightweight supervised finetuning, the training
 2207 time of which ranges from several minutes to two hours at most.

2208 Compared to the Base prior, those specifically enhanced with Agentless skills (SFT and RL) demon-
 2209 strate stronger task generalization, especially under the data-scarce settings. However, when more
 2210 SWE-Smith trajectories are used for adaptation, the performances of the Base and the MT priors
 2211 become closer to those of the SFT and the RL priors. This could be attributed to the gaps between
 2212 the different SWE tasks. The exploration for recipes that enable stronger out-of-distribution and
 2213 task-agnostic generalization is left for future work.

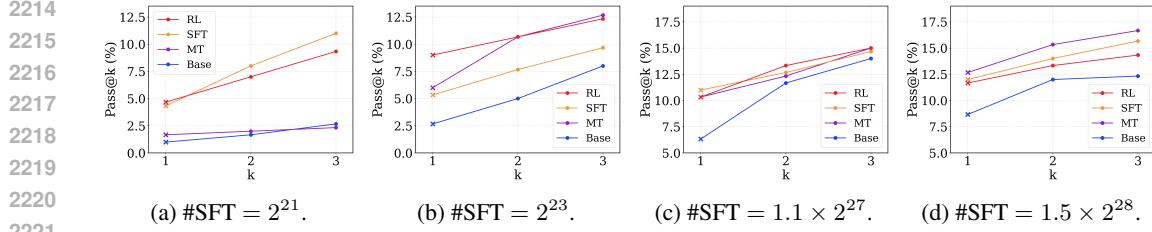


Figure 10: Generalization analysis on SWE-bench-Live.

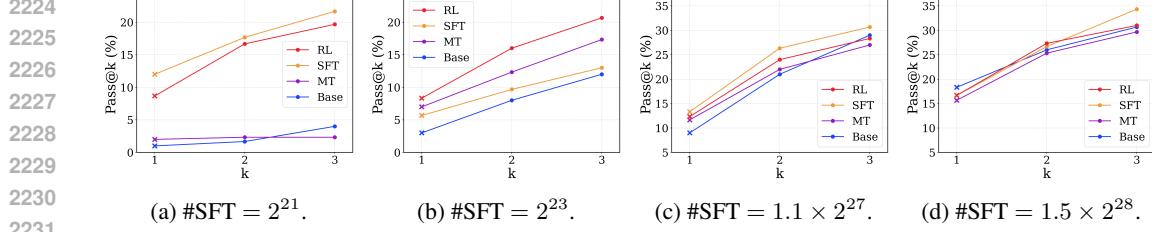


Figure 11: Generalization analysis on SWE-bench Multilingual.

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.

2243
2244¹⁰⁵¹ <think>
2245 Okay, let's tackle this problem. The user reported an issue where the
2246 `Matrix.col_insert()` method isn't working correctly. The example
2247 given shows that after inserting a matrix `V` into `M` at position 3,
2248 the resulting matrix has the inserted columns shifted incorrectly.
2249 The expected result would have the inserted columns at positions 3
2250 and 4, but the actual output shows them starting at position 3, but
2251 the original matrix's columns after the insert are shifted upwards
1053 instead of remaining in their original positions.
2252¹⁰⁵⁴ First, I need to understand how `col_insert` works. Looking at the code
2253 provided, especially in the `MatrixShaping` class in `sympy/matrices/`
2254 `common.py`, the `_eval_col_insert` method is responsible for inserting
2255 columns. The current implementation shifts the existing columns to
2256 the right by the number of columns inserted. However, the bug might
2257 be in how the original matrix's columns are being shifted.
2258¹⁰⁵⁶ Let me look at the existing code for `_eval_col_insert`:
2259¹⁰⁵⁷
2260¹⁰⁵⁸ `def _eval_col_insert(self, pos, other):`
2261¹⁰⁵⁹ `cols = self.cols`
2262¹⁰⁶⁰
2263¹⁰⁶¹ `def entry(i, j):`
2264¹⁰⁶² `if j < pos:`
2265¹⁰⁶³ `return self[i, j]`
2266¹⁰⁶⁴ `elif pos <= j < pos + other.cols:`
2267¹⁰⁶⁵ `return other[i, j - pos]`
2268¹⁰⁶⁶ `return self[i, j - pos - other.cols]`
2269¹⁰⁶⁷
2270¹⁰⁶⁸ `return self._new(self.rows, self.cols + other.cols,`

```

2268          lambda i, j: entry(i, j))

22691069
22701070 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.

22781072
22791073 Wait, let's think again. The original code for the entry function:

22801075
22811076 if j < pos:
22821077     return self[i, j] # original columns before pos
22831078 elif pos <= j < pos + other.cols:
22841079     return other[i, j - pos] # inserted columns
22851080 return self[i, j - pos - other.cols] # original columns after the
22861081      inserted ones

22861081 Wait, no. Let's say the original matrix has cols = 6. We insert 2
2287      columns at pos=3. The new total cols is 6 + 2 = 8. The original
2288      columns after pos 3 (columns 3,4,5) should now be at positions
2289      5,6,7. But according to the code, when j is >=3 +2=5 (since pos +
2290      other.cols =3+2=5), then the code returns self[i, j -5 -2] = self[i,
2291      j -7]. Wait, that's not right. For example, if j=5 (the new column
2292      5), then j - pos - other.cols =5 -3 -2=0. So the original column 0
22931082      would be mapped to the new column 5. That's definitely wrong.

22931082
22941083 Ah, here's the mistake. The code is subtracting (pos + other.cols) from
2295      j, but that's incorrect. The original columns after the inserted
2296      ones should be shifted to the right by other.cols. So, the correct
2297      calculation should be j - other.cols. For example, original column 3
22981084      becomes 3 +2=5, original column 4 becomes 6, etc.

22981084
22991085 So the line in the code:

23001087 return self[i, j - pos - other.cols]

23011088
23021089 should be:

23031090
23041091 return self[i, j - other.cols]

23051093 Because the inserted columns have taken up 'other.cols' positions
2306      starting at 'pos', so the original columns after 'pos' are shifted
2307      by 'other.cols' to the right. Therefore, to get the correct original
2308      column index, we subtract 'other.cols' from the current j.

23081094
23091095 So the fix is to change the subtraction from (pos + other.cols) to
2310      other.cols. Let me check:

23111096
23121097 Original code:

23131098
23141099 return self[i, j - pos - other.cols]

23151101
23161102 Corrected code:

23171103
23181104 return self[i, j - other.cols]

23181104
23191105 Wait, no. Wait, the original columns after the inserted block (which
2320      starts at pos and has other.cols columns) would start at j = pos +
2321      other.cols. So for those j values, the original column index would
2322      be j - (pos + other.cols). Wait, no. Let's think with numbers.

1106

```

2322¹¹⁰⁷ Original matrix has 6 columns. Insert 2 columns at position 3. The new
 2323 matrix has 8 columns. The original columns 0,1,2 remain in positions
 2324 0,1,2. The inserted 2 columns are at positions 3 and 4. The
 2325 original columns 3,4,5 now become 5,6,7.
 2326¹¹⁰⁸
 2327¹¹⁰⁹ So for j in 5,6,7 (original columns 3,4,5), the mapping is:
 2328¹¹¹⁰
 2329¹¹¹¹ 5 \rightarrow 3
 2330¹¹¹² 6 \rightarrow 4
 2331¹¹¹³ 7 \rightarrow 5
 2332¹¹¹⁴
 2333¹¹¹⁵ So the formula should be $original_column = j - inserted_columns$. In
 2334 this case, $inserted_columns$ is $other.cols$. So the code should
 2335 subtract $other.cols$ from j when accessing the original matrix's
 2336 columns after the inserted block.
 2337¹¹¹⁶
 2338¹¹¹⁷ Therefore, the line:
 2339¹¹¹⁸
 2340¹¹¹⁹ return $self[i, j - pos - other.cols]$
 2341¹¹²⁰
 2342¹¹²¹ should be:
 2343¹¹²²
 2344¹¹²³ return $self[i, j - other.cols]$
 2345 Because pos is where the insertion starts, but the original columns
 2346 after the inserted block are shifted by $other.cols$. Let's verify
 2347 with the example:
 2348¹¹²⁴
 2349¹¹²⁵ $pos = 3$, $other.cols = 2$. For $j=5$ (new column 5), original column index is
 2350¹¹²⁶ 5 - 2 = 3, which is correct. For $j=6$, 6 - 2 = 4; $j=7$, 7 - 2 = 5. That's correct
 2351¹¹²⁷.
 2352 So the error in the original code was subtracting $(pos + other.cols)$
 2353 instead of $other.cols$. The pos is the starting position of the
 2354 insertion, but after inserting $other.cols$ columns, the original
 2355 columns after the inserted block are shifted by $other.cols$.
 2356 Therefore, the correct subtraction is $other.cols$.
 2357¹¹²⁸
 2358¹¹²⁹ Thus, the fix is to change the line in the $_eval_col_insert$ method from
 2359¹¹³⁰ :
 2360¹¹³¹
 2361¹¹³² return $self[i, j - pos - other.cols]$
 2362¹¹³³
 2363¹¹³⁴ to:
 2364¹¹³⁵
 2365¹¹³⁶ return $self[i, j - other.cols]$
 2366¹¹³⁷
 2367¹¹³⁸ Now, looking back at the code in the $_eval_col_insert$ function:
 2368¹¹³⁹
 2369¹¹⁴⁰ The code for the $else$ clause is:
 2370¹¹⁴¹
 2371¹¹⁴² return $self[i, j - pos - other.cols]$
 2372¹¹⁴³
 2373¹¹⁴⁴ Which is incorrect. Changing that to $j - other.cols$ should fix the
 2374¹¹⁴⁵ problem.
 2375¹¹⁴⁶
 2376¹¹⁴⁷ So the SEARCH block would look for the lines:
 2377¹¹⁴⁸
 2378¹¹⁴⁹ def $_eval_col_insert(self, pos, other):$
 2379¹¹⁵⁰ cols = $self.cols$
 2380¹¹⁵¹
 2381¹¹⁵² def entry(i, j):
 2382¹¹⁵³ if $j < pos$:
 2383¹¹⁵⁴
 2384¹¹⁵⁵

```

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))

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]

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
23921174 Therefore, the correct SEARCH/REPLACE block would be:
23931175
23941176 ##### sympy/matrices/common.py
23951177 <<<<< SEARCH
23961178     def _eval_col_insert(self, pos, other):
23971179         cols = self.cols

23981180
23991181         def entry(i, j):
24001182             if j < pos:
24011183                 return self[i, j]
24021184             elif pos <= j < pos + other.cols:
24031185                 return other[i, j - pos]
24041186             return self[i, j - pos - other.cols]

24051187
24061188         return self._new(self.rows, self.cols + other.cols,
24071189                         lambda i, j: entry(i, j))
24081190 =====
24091191     def _eval_col_insert(self, pos, other):
24101192         def entry(i, j):
24111193             if j < pos:
24121194                 return self[i, j]
24131195             elif pos <= j < pos + other.cols:
24141196                 return other[i, j - pos]
24151197             return self[i, j - other.cols]

24161198
24171199         return self._new(self.rows, self.cols + other.cols,
24181200                         lambda i, j: entry(i, j))
24191201 >>>>> REPLACE
24201202
24211203 This change corrects the column shifting calculation for the original
24221204     matrix's columns after the inserted block. Instead of subtracting ( 
24231205     pos + other.cols), we subtract other.cols, which properly shifts the
24241206     original columns to the right by the number of inserted columns.
24251207 </think>
24261208
24271209     '''python
24281210 ##### sympy/matrices/common.py
24291211 <<<<< SEARCH
24301212     def _eval_col_insert(self, pos, other):
24311213         cols = self.cols

24321214
24331215         def entry(i, j):
24341216             if j < pos:
24351217                 return self[i, j]
24361218             elif pos <= j < pos + other.cols:
24371219                 return other[i, j - pos]
24381220             return self[i, j - pos - other.cols]

```

```
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
2442 1228         return self._new(self.rows, self.cols + other.cols,
2443 1229                 lambda i, j: entry(i, j))
2444 1230 >>>>> REPLACE
2445 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) ***) == TP(A***, B***)`. As a consequence, it fails to serve as a reliable judge for BugFixer.

```
2464  
2465 1 diff --git a/tests/template_tests/filter_tests/test_escapingseq.py b/  
2466     tests/template_tests/filter_tests/test_escapingseq.py  
2467 2 new file mode 100644  
2468 3 --- /dev/null  
2469 4 +++ b/tests/template_tests/filter_tests/test_escapingseq.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 EscapingseqTests(SimpleTestCase):  
2478 13 +    """  
2479 14 +        The "escapingseq" 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 +    @setup(  
2484 19 +        {  
2485 20 +            "escapingseq_basic": (  
2486 21 +                '{{ a|escapingseq|join:", " }} -- {{ b|escapingseq|join:",  
2487 22 +                " }}',
```

```

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("<p>")]},
2491 29       )
2492 30       self.assertEqual(output, "x&y, <p> -- x&y, <p>")
2493 31   +
2494 32   @setup(
2495 33       {
2496 34           "escapeseq_autoescape_off": (
2497 35               '{% autoescape off %}{{ a|escapeseq|join:", " }}'
2498 36               " -- "
2499 37               '{{ b|escapeseq|join:", " }}{{ endautoescape %}'
2500 38           )
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("<p>")]},
2507 45       )
2508 46       self.assertEqual(output, "x&y, <p> -- x&y, <p>")
2509 47   +
2510 48   @setup({"escapeseq_join": '{{ a|escapeseq|join:"<br/>" }}'})
2511 49   def test_chain_join(self):
2512 50       output = self.engine.render_to_string("escapeseq_join", {"a": [
2513 51           ["x&y", "<p>"]
2514 52       ]})
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/>" }}{{ endautoescape %}'
2521 59           ),
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         # New tests for escapeseq
2536 11         @setup({"escapeseq01": '{{ a|escapeseq|join:", " }}'})
2537 12         def test_escaping01(self):
2538             output = self.engine.render_to_string("escapeseq01", {"a": [
2539                 "&", "<"}})

```

```

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_escaping02(self):
2548 23         output = self.engine.render_to_string("escapeseq02", {"a":'
2549 24             ["&", "<"]})
2550 25         self.assertEqual(output, "&lt;")

```

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

2552

```

2553 1 diff --git a/sympy/physics/quantum/tests/test_tensorproduct.py b/sympy/
2554 2     physics/quantum/tests/test_tensorproduct.py
2555 3 --- a/sympy/physics/quantum/tests/test_tensorproduct.py
2556 4 +++ b/sympy/physics/quantum/tests/test_tensorproduct.py
2557 5 @@ -10,7 +10,7 @@
2558 6     from sympy.physics.quantum.density import Density
2559 7     from sympy.core.trace import Tr
2560 8     -A, B, C = symbols('A,B,C', commutative=False)
2561 9     +A, B, C, D = symbols('A,B,C,D', commutative=False)
2562 10    x = symbols('x')
2563 11
2564 12    mat1 = Matrix([[1, 2*I], [1 + I, 3]])
2565 13 @@ -47,6 +47,11 @@
2566 14    def test_tensor_product_commutator():
2567 15
2568 16        def test_tensor_product_simp():
2569 17            assert tensor_product_simp(TP(A, B)*TP(B, C)) == TP(A*B, B*C)
2570 18            # tests for Pow-expressions
2571 19            assert tensor_product_simp(TP(A, B)**x) == TP(A**x, B**x)
2572 20            assert tensor_product_simp(x*TP(A, B)**2) == x*TP(A**2, B**2)
2573 21            assert tensor_product_simp(x*(TP(A, B)**2)*TP(C, D)) == x*TP(A**2*C
2574 22            , B**2*D)
2575 23            assert tensor_product_simp(TP(A, B)-TP(C, D)**x) == TP(A, B)-TP(C*x,
2576 24            D*x)
2577
2578 25    def test_issue_5923():
2579
2580 26
2581 27
2582 28
2583 29
2584 30
2585 31
2586 32
2587 33
2588 34
2589 35
2590 36
2591 37
2592 38
2593 39
2594 40
2595 41
2596 42
2597 43
2598 44
2599 45
2600 46
2601 47
2602 48
2603 49
2604 50
2605 51
2606 52
2607 53
2608 54
2609 55
2610 56
2611 57
2612 58
2613 59
2614 60
2615 61
2616 62
2617 63
2618 64
2619 65
2620 66
2621 67
2622 68
2623 69
2624 70
2625 71
2626 72
2627 73
2628 74
2629 75
2630 76
2631 77
2632 78
2633 79
2634 80
2635 81
2636 82
2637 83
2638 84
2639 85
2640 86
2641 87
2642 88
2643 89
2644 90
2645 91
2646 92
2647 93
2648 94
2649 95
2650 96
2651 97
2652 98
2653 99
2654 100
2655 101
2656 102
2657 103
2658 104
2659 105
2660 106
2661 107
2662 108
2663 109
2664 110
2665 111
2666 112
2667 113
2668 114
2669 115
2670 116
2671 117
2672 118
2673 119
2674 120
2675 121
2676 122
2677 123
2678 124
2679 125
2680 126
2681 127
2682 128
2683 129
2684 130
2685 131
2686 132
2687 133
2688 134
2689 135
2690 136
2691 137
2692 138
2693 139
2694 140
2695 141
2696 142
2697 143
2698 144
2699 145
2700 146
2701 147
2702 148
2703 149
2704 150
2705 151
2706 152
2707 153
2708 154
2709 155
2710 156
2711 157
2712 158
2713 159
2714 160
2715 161
2716 162
2717 163
2718 164
2719 165
2720 166
2721 167
2722 168
2723 169
2724 170
2725 171
2726 172
2727 173
2728 174
2729 175
2730 176
2731 177
2732 178
2733 179
2734 180
2735 181
2736 182
2737 183
2738 184
2739 185
2740 186
2741 187
2742 188
2743 189
2744 190
2745 191
2746 192
2747 193
2748 194
2749 195
2750 196
2751 197
2752 198
2753 199
2754 200
2755 201
2756 202
2757 203
2758 204
2759 205
2760 206
2761 207
2762 208
2763 209
2764 210
2765 211
2766 212
2767 213
2768 214
2769 215
2770 216
2771 217
2772 218
2773 219
2774 220
2775 221
2776 222
2777 223
2778 224
2779 225
2780 226
2781 227
2782 228
2783 229
2784 230
2785 231
2786 232
2787 233
2788 234
2789 235
2790 236
2791 237
2792 238
2793 239
2794 240
2795 241
2796 242
2797 243
2798 244
2799 245
2800 246
2801 247
2802 248
2803 249
2804 250
2805 251
2806 252
2807 253
2808 254
2809 255
2810 256
2811 257
2812 258
2813 259
2814 260
2815 261
2816 262
2817 263
2818 264
2819 265
2820 266
2821 267
2822 268
2823 269
2824 270
2825 271
2826 272
2827 273
2828 274
2829 275
2830 276
2831 277
2832 278
2833 279
2834 280
2835 281
2836 282
2837 283
2838 284
2839 285
2840 286
2841 287
2842 288
2843 289
2844 290
2845 291
2846 292
2847 293
2848 294
2849 295
2850 296
2851 297
2852 298
2853 299
2854 300
2855 301
2856 302
2857 303
2858 304
2859 305
2860 306
2861 307
2862 308
2863 309
2864 310
2865 311
2866 312
2867 313
2868 314
2869 315
2870 316
2871 317
2872 318
2873 319
2874 320
2875 321
2876 322
2877 323
2878 324
2879 325
2880 326
2881 327
2882 328
2883 329
2884 330
2885 331
2886 332
2887 333
2888 334
2889 335
2890 336
2891 337
2892 338
2893 339
2894 340
2895 341
2896 342
2897 343
2898 344
2899 345
2900 346
2901 347
2902 348
2903 349
2904 350
2905 351
2906 352
2907 353
2908 354
2909 355
2910 356
2911 357
2912 358
2913 359
2914 360
2915 361
2916 362
2917 363
2918 364
2919 365
2920 366
2921 367
2922 368
2923 369
2924 370
2925 371
2926 372
2927 373
2928 374
2929 375
2930 376
2931 377
2932 378
2933 379
2934 380
2935 381
2936 382
2937 383
2938 384
2939 385
2940 386
2941 387
2942 388
2943 389
2944 390
2945 391
2946 392
2947 393
2948 394
2949 395
2950 396
2951 397
2952 398
2953 399
2954 400
2955 401
2956 402
2957 403
2958 404
2959 405
2960 406
2961 407
2962 408
2963 409
2964 410
2965 411
2966 412
2967 413
2968 414
2969 415
2970 416
2971 417
2972 418
2973 419
2974 420
2975 421
2976 422
2977 423
2978 424
2979 425
2980 426
2981 427
2982 428
2983 429
2984 430
2985 431
2986 432
2987 433
2988 434
2989 435
2990 436
2991 437
2992 438
2993 439
2994 440
2995 441
2996 442
2997 443
2998 444
2999 445
2999 446
2999 447
2999 448
2999 449
2999 450
2999 451
2999 452
2999 453
2999 454
2999 455
2999 456
2999 457
2999 458
2999 459
2999 460
2999 461
2999 462
2999 463
2999 464
2999 465
2999 466
2999 467
2999 468
2999 469
2999 470
2999 471
2999 472
2999 473
2999 474
2999 475
2999 476
2999 477
2999 478
2999 479
2999 480
2999 481
2999 482
2999 483
2999 484
2999 485
2999 486
2999 487
2999 488
2999 489
2999 490
2999 491
2999 492
2999 493
2999 494
2999 495
2999 496
2999 497
2999 498
2999 499
2999 500
2999 501
2999 502
2999 503
2999 504
2999 505
2999 506
2999 507
2999 508
2999 509
2999 510
2999 511
2999 512
2999 513
2999 514
2999 515
2999 516
2999 517
2999 518
2999 519
2999 520
2999 521
2999 522
2999 523
2999 524
2999 525
2999 526
2999 527
2999 528
2999 529
2999 530
2999 531
2999 532
2999 533
2999 534
2999 535
2999 536
2999 537
2999 538
2999 539
2999 540
2999 541
2999 542
2999 543
2999 544
2999 545
2999 546
2999 547
2999 548
2999 549
2999 550
2999 551
2999 552
2999 553
2999 554
2999 555
2999 556
2999 557
2999 558
2999 559
2999 560
2999 561
2999 562
2999 563
2999 564
2999 565
2999 566
2999 567
2999 568
2999 569
2999 570
2999 571
2999 572
2999 573
2999 574
2999 575
2999 576
2999 577
2999 578
2999 579
2999 580
2999 581
2999 582
2999 583
2999 584
2999 585
2999 586
2999 587
2999 588
2999 589
2999 590
2999 591
2999 592
2999 593
2999 594
2999 595
2999 596
2999 597
2999 598
2999 599
2999 600
2999 601
2999 602
2999 603
2999 604
2999 605
2999 606
2999 607
2999 608
2999 609
2999 610
2999 611
2999 612
2999 613
2999 614
2999 615
2999 616
2999 617
2999 618
2999 619
2999 620
2999 621
2999 622
2999 623
2999 624
2999 625
2999 626
2999 627
2999 628
2999 629
2999 630
2999 631
2999 632
2999 633
2999 634
2999 635
2999 636
2999 637
2999 638
2999 639
2999 640
2999 641
2999 642
2999 643
2999 644
2999 645
2999 646
2999 647
2999 648
2999 649
2999 650
2999 651
2999 652
2999 653
2999 654
2999 655
2999 656
2999 657
2999 658
2999 659
2999 660
2999 661
2999 662
2999 663
2999 664
2999 665
2999 666
2999 667
2999 668
2999 669
2999 670
2999 671
2999 672
2999 673
2999 674
2999 675
2999 676
2999 677
2999 678
2999 679
2999 680
2999 681
2999 682
2999 683
2999 684
2999 685
2999 686
2999 687
2999 688
2999 689
2999 690
2999 691
2999 692
2999 693
2999 694
2999 695
2999 696
2999 697
2999 698
2999 699
2999 700
2999 701
2999 702
2999 703
2999 704
2999 705
2999 706
2999 707
2999 708
2999 709
2999 710
2999 711
2999 712
2999 713
2999 714
2999 715
2999 716
2999 717
2999 718
2999 719
2999 720
2999 721
2999 722
2999 723
2999 724
2999 725
2999 726
2999 727
2999 728
2999 729
2999 730
2999 731
2999 732
2999 733
2999 734
2999 735
2999 736
2999 737
2999 738
2999 739
2999 740
2999 741
2999 742
2999 743
2999 744
2999 745
2999 746
2999 747
2999 748
2999 749
2999 750
2999 751
2999 752
2999 753
2999 754
2999 755
2999 756
2999 757
2999 758
2999 759
2999 760
2999 761
2999 762
2999 763
2999 764
2999 765
2999 766
2999 767
2999 768
2999 769
2999 770
2999 771
2999 772
2999 773
2999 774
2999 775
2999 776
2999 777
2999 778
2999 779
2999 780
2999 781
2999 782
2999 783
2999 784
2999 785
2999 786
2999 787
2999 788
2999 789
2999 790
2999 791
2999 792
2999 793
2999 794
2999 795
2999 796
2999 797
2999 798
2999 799
2999 800
2999 801
2999 802
2999 803
2999 804
2999 805
2999 806
2999 807
2999 808
2999 809
2999 810
2999 811
2999 812
2999 813
2999 814
2999 815
2999 816
2999 817
2999 818
2999 819
2999 820
2999 821
2999 822
2999 823
2999 824
2999 825
2999 826
2999 827
2999 828
2999 829
2999 830
2999 831
2999 832
2999 833
2999 834
2999 835
2999 836
2999 837
2999 838
2999 839
2999 840
2999 841
2999 842
2999 843
2999 844
2999 845
2999 846
2999 847
2999 848
2999 849
2999 850
2999 851
2999 852
2999 853
2999 854
2999 855
2999 856
2999 857
2999 858
2999 859
2999 860
2999 861
2999 862
2999 863
2999 864
2999 865
2999 866
2999 867
2999 868
2999 869
2999 870
2999 871
2999 872
2999 873
2999 874
2999 875
2999 876
2999 877
2999 878
2999 879
2999 880
2999 881
2999 882
2999 883
2999 884
2999 885
2999 886
2999 887
2999 888
2999 889
2999 890
2999 891
2999 892
2999 893
2999 894
2999 895
2999 896
2999 897
2999 898
2999 899
2999 900
2999 901
2999 902
2999 903
2999 904
2999 905
2999 906
2999 907
2999 908
2999 909
2999 910
2999 911
2999 912
2999 913
2999 914
2999 915
2999 916
2999 917
2999 918
2999 919
2999 920
2999 921
2999 922
2999 923
2999 924
2999 925
2999 926
2999 927
2999 928
2999 929
2999 930
2999 931
2999 932
2999 933
2999 934
2999 935
2999 936
2999 937
2999 938
2999 939
2999 940
2999 941
2999 942
2999 943
2999 944
2999 945
2999 946
2999 947
2999 948
2999 949
2999 950
2999 951
2999 952
2999 953
2999 954
2999 955
2999 956
2999 957
2999 958
2999 959
2999 960
2999 961
2999 962
2999 963
2999 964
2999 965
2999 966
2999 967
2999 968
2999 969
2999 970
2999 971
2999 972
2999 973
2999 974
2999 975
2999 976
2999 977
2999 978
2999 979
2999 980
2999 981
2999 982
2999 983
2999 984
2999 985
2999 986
2999 987
2999 988
2999 989
2999 990
2999 991
2999 992
2999 993
2999 994
2999 995
2999 996
2999 997
2999 998
2999 999
2999 1000
2999 1001
2999 1002
2999 1003
2999 1004
2999 1005
2999 1006
2999 1007
2999 1008
2999 1009
2999 1010
2999 1011
2999 1012
2999 1013
2999 1014
2999 1015
2999 1016
2999 1017
2999 1018
2999 1019
2999 1020
2999 1021
2999 1022
2999 1023
2999 1024
2999 1025
2999 1026
2999 1027
2999 1028
2999 1029
2999 1030
2999 1031
2999 1032
2999 1033
2999 1034
2999 1035
2999 1036
2999 1037
2999 1038
2999 1039
2999 1040
2999 1041
2999 1042
2999 1043
2999 1044
2999 1045
2999 1046
2999 1047
2999 1048
2999 1049
2999 1050
2999 1051
2999 1052
2999 1053
2999 1054
2999 1055
2999 1056
2999 1057
2999 1058
2999 1059
2999 1060
2999 1061
2999 1062
2999 1063
2999 1064
2999 1065
2999 1066
2999 1067
2999 1068
2999 1069
2999 1070
2999 1071
2999 1072
2999 1073
2999 1074
2999 1075
2999 1076
2999 1077
2999 1078
2999 1079
2999 1080
2999 1081
2999 1082
2999 1083
2999 1084
2999 1085
2999 1086
2999 1087
2999 1088
2999 1089
2999 1090
2999 1091
2999 1092
2999 1093
2999 1094
2999 1095
2999 1096
2999 1097
2999 1098
2999 1099
2999 1100
2999 1101
2999 1102
2999 1103
2999 1104
2999 1105
2999 1106
2999 1107
2999 1108
2999 1109
2999 1110
2999 1111
2999 1112
2999 1113
2999 1114
2999 1115
2999 1116
2999 1117
2999 1118
2999 1119
2999 1120
2999 1121
2999 1122
2999 1123
2999 1124
2999 1125
2999 1126
2999 1127
2999 1128
2999 1129
2999 1130
2999 1131
2999 1132
2999 1133
2999 1134
2999 1135
2999 1136
2999 1137
2999 1138
2999 1139
2999 1140
2999 1141
2999 1142
2999 1143
2999 1144
2999 1145
2999 1146
2999 1147
2999 1148
2999 1149
2999 1150
2999 1151
2999 1152
2999 1153
2999 1154
2999 1155
2999 1156
2999 1157
2999 1158
2999 1159
2999 1160
2999 1161
2999 1162
2999 1163
2999 1164
2999 1165
2999 1166
2999 1167
2999 1168
2999 1169
2999 1170
2999 1171
2999 1172
2999 1173
2999 1174
2999 1175
2999 1176

```

```

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 +
2602 27 +     expr2 = TP(1, Pauli(3)) * TP(1, a)
2603 28 +     result2 = expr2.subs(a, Pauli(3))
2604 29 +     assert tensor_product_simp(result2) == TP(1, 1), f"Substitution 2
2605 30 failed: {result2}"
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.

```

26271232 We are currently solving the following issue within our repository.
26281233 Here is the issue text:
26291234 --- BEGIN ISSUE ---
26301235 {problem_statement}
26311236 --- END ISSUE ---

26321237 Below are some thinking trajectories, each from llm reasoning model.
26331238 Only one trajectory is right.
26341239 --- BEGIN FILE ---
26351240 ``
26361241 {trajs_content}
26371242 ``
26381243 --- END FILE ---

26391244 Please first summary and analyze the key differences between the
26401245 trajectories, and then generate *SEARCH/REPLACE* edits to fix the
26411246 issue.

26421247 Every *SEARCH/REPLACE* edit must use this format:
26431248 1. The file path
26441249 2. The start of search block: <<<<< SEARCH
26451250 3. A contiguous chunk of lines to search for in the existing source
code
4. The dividing line: =====

```

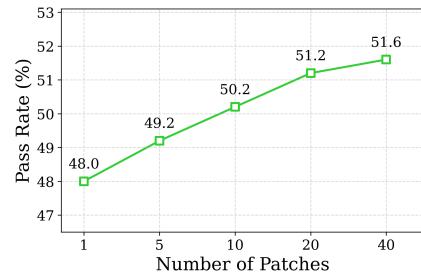


Figure 12: Emergent parallel scaling results on SWE-bench Verified.

```

26461251 5. The lines to replace into the source code
26471252 6. The end of the replace block: >>>>> REPLACE
26481253
26491254 Here is an example:
26501255
26511256 '''python
26521257 ### mathweb/flask/app.py
26531258 <<<<< SEARCH
26541259 from flask import Flask
26551260 =====
26561261 import math
26571262 from flask import Flask
26581263 >>>>> REPLACE
26591264 '''
26601265
26611266 Please note that the *SEARCH/REPLACE* edit REQUIRES PROPER INDENTATION.
26621267 If you would like to add the line ' print(x)', you must
26631268 fully write that out, with all those spaces before the code!
26641269 Wrap the *SEARCH/REPLACE* edit in blocks '''python...'''.
26651270 The summary of the key differences between the trajectories should be
26661271 in the thinking part.
26671272
26681273
26691274
26701275
26711276
26721277
26731278
26741279
26751280
26761281
26771282
26781283
26791284
26801285
26811286
26821287
26831288
26841289
26851290
26861291
26871292
26881293
26891294

```

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.

```

26731269 Prompt:
26741270 We are currently solving the following issue within our repository.
26751271 Here is the issue text:
26761272 --- BEGIN ISSUE ---
26771273 Data <@filename isn't relative to the YAML file
26781274 The [docs say] (https://gabbi.readthedocs.io/en/latest/format.html#data)
26791275 :
26801276 > If the value is a string that begins with <@ then the rest of the
26811277 string is treated as the name of a file to be loaded from the same
26821278 directory as the YAML file.
26831279 But I haven't found it works like this unless I cd into the directory
26841280 containing the yaml file.
26851281
26861282 '_load_data_file' says:
26871283 '''python
26881284 path = os.path.join(self.test_directory, os.path.basename(filename))
26891285 '''
26901286 this does a few things:
26911287 - makes the path begin with 'self.test_directory' (defaults to '.', the
26921288 current working directory)
26931289 - disregards any folders in the <@ 'filename' path
26941290 - appends 'filename'
26951291
26961292 This means, if I'm in '/', and I have a test that says 'data: <@cake.
26971293 jpg', I cannot run my tests as advised in the docs:
26981294 'gabbi-run -- /my/test.yaml /my/other.yaml'
26991295
27001296 'FileNotFoundError: [Errno 2] No such file or directory: './cake.jpg'
27011297
27021298 So in our test running script, we have to start with 'cd tests/'.
27031299
27041300

```

2700¹²⁹⁵ My preference, would be to make the path relative to the location of
 2701¹²⁹⁶ the yaml file. That way, I can run my tests from whatever directory.
 2702¹²⁹⁶
 2703¹²⁹⁷ Obviously this isn't possible when redirecting with '<' as the filename
 2704¹²⁹⁸ is never seen by gabbi. So I'm happy to discuss other ideas.
 2705¹²⁹⁸
 2706¹²⁹⁹
 2707¹³⁰⁰
 2708¹³⁰¹ Furthermore I can't keep my test data in a child directory, unless I cd
 into that directory like:
 2709¹³⁰² ```shell
 2710¹³⁰³ cd yaml_tests/test_data
 2711¹³⁰⁴ gabbi-run -- ./test_things.yaml
 2712¹³⁰⁵ ````
 2713¹³⁰⁷ So for that reason, I'd like to allow directories like '<@test_data/
 2714¹³⁰⁸ filename.txt' to be included in the path.
 2715¹³⁰⁸
 2716¹³⁰⁹
 2717¹³¹⁰
 2718¹³¹¹ Happy to write the patch, if we come up with an approach here.
 2719¹³¹² --- END ISSUE ---
 2720¹³¹⁴
 2721¹³¹⁵
 2722¹³¹⁷ Below are some thinking trajectories, each from llm reasoning model.
 2723¹³¹⁸
 2724¹³¹⁹ --- BEGIN FILE ---
 2725¹³²⁰ ````
Answer 1:
 2726¹³²¹ ```python
 2727¹³²² #### gabbi/runner.py
 2728¹³²³ <<<<< SEARCH
 2729¹³²⁴ import argparse
 2730¹³²⁵ from importlib import import_module
 2731¹³²⁶ import sys
 2732¹³²⁷ import unittest
 2733¹³²⁸ =====

2734¹³²⁹ import argparse
 2735¹³³⁰ from importlib import import_module
 2736¹³³¹ import os
 2737¹³³² import sys
 2738¹³³³ import unittest
 2739¹³³⁴ >>>>> REPLACE
 2740¹³³⁵ ````
 2741¹³³⁶ ```python
 2742¹³³⁷ #### gabbi/runner.py
 2743¹³³⁸ <<<<< SEARCH
 2744¹³³⁹ def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
 2745¹³⁴⁰ False,
 2746¹³⁴¹ failfast=False):
 2747¹³⁴² """Run the tests from the YAML in handle."""
 2748¹³⁴³ data = utils.load_yaml(handle)
 2749¹³⁴⁴ if force_ssl:
 2750¹³⁴⁵ if 'defaults' in data:
 2751¹³⁴⁶ data['defaults']['ssl'] = True
 2752¹³⁴⁷ else:
 2753¹³⁴⁸ data['defaults'] = {'ssl': True}
 2754¹³⁴⁹ loader = unittest.defaultTestLoader
 2755¹³⁵⁰ test_suite = suitemaker.test_suite_from_dict(
 2756¹³⁵¹ loader, 'input', data, '.', host, port, None, None, prefix=
 2757¹³⁵² prefix,
 2758¹³⁵³ handlers=handler_objects)
 2759¹³⁵⁴ result = ConciseTestRunner(
 2760¹³⁵⁵)

```

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 = suitemaker.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
Answer 2:
2797 '''python
2798 ### gabbi/runner.py
2799 <<<<< SEARCH
2800 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2801     False,
2802     failfast=False):
2803 =====
2804 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2805     False,
2806     failfast=False, test_directory='.'):
2807 >>>>> REPLACE
2808 """
2809 '''python
2810 ### gabbi/runner.py
2811 <<<<< SEARCH
2812     test_suite = suitemaker.test_suite_from_dict(
2813         loader, 'input', data, '.', host, port, None, None, prefix=
2814             prefix,
2815             handlers=handler_objects)
2816 =====
2817     test_suite = suitemaker.test_suite_from_dict(
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
3999
4000
4000
4001
4001
4002
4002
4003
4003
4004
4004
4005
4005
4006
4006
4007
4007
4008
4008
4009
4009
4010
4010
4011
4011
4012
4012
4013
4013
4014
4014
4015
4015
4016
4016
4017
4017
4018
4018
4019
4019
4020
4020
4021
4021
4022
4022
4023
4023
4024
4024
4025
4025
4026
4026
4027
4027
4028
4028
4029
4029
4030
4030
4031
4031
4032
4032
4033
4033
4034
4034
4035
4035
4036
4036
4037
4037
4038
4038
4039
4039
4040
4040
4041
4041
4042
4042
4043
4043
4044
4044
4045
4045
4046
4046
4047
4047
4048
4048
4049
4049
4050
4050
4051
4051
4052
4052
4053
4053
4054
4054
4055
4055
4056
4056
4057
4057
4058
4058
4059
4059
4060
4060
4061
4061
4062
4062
4063
4063
4064
4064
4065
4065
4066
4066
4067
4067
4068
4068
4069
4069
4070
4070
4071
4071
4072
4072
4073
4073
4074
4074
4075
4075
4076
4076
4077
4077
4078
4078
4079
4079
4080
4080
4081
4081
4082
4082
4083
4083
4084
4084
4085
4085
4086
4086
4087
4087
4088
4088
4089
4089
4090
4090
4091
4091
4092
4092
4093
4093
4094
4094
4095
4095
4096
4096
4097
4097
4098
4098
4099
4099
4100
4100
4101
4101
4102
4102
4103
4103
4104
4104
4105
4105
4106
4106
4107
4107
4108
4108
4109
4109
4110
4110
4111
4111
4112
4112
4113
4113
4114
4114
4115
4115
4116
4116
4117
4117
4118
4118
4119
4119
4120
4120
4121
4121
4122
4122
4123
4123
4124
4124
4125
4125
4126
4126
4127
4127
4128
4128
4129
4129
4130
4130
4131
4131
4132
4132
4133
4133
4134
4134
4135
4135
4136
4136
4137
4137
4138
4138
4139
4139
4140
4140
4141
4141
4142
4142
4143
4143
4144
4144
4145
4145
4146
4146
4147
4147
4148
4148
4149
4149
4150
4150
4151
4151
4152
4152
4153
4153
4154
4154
4155
4155
4156
4156
4157
4157
4158
4158
4159
4159
4160
4160
4161
4161
4162
4162
4163
4163
4164
4164
4165
4165
4166
4166
4167
4167
4168
4168
4169
4169
4170
4170
4171
4171
4172
4172
4173
4173
4174
4174
4175
4175
4176
4176
4177
4177
4178
4178
4179
4179
4180
4180
4181
4181
4182
4182
4183
4183
4184
4184
4185
4185
4186
4186
4187
4187
4188
4188
4189
4189
4190
4190
4191
4191
4192
4192
4193
4193
4194
4194
4195
4195
4196
4196
4197
4197
4198
4198
4199
4199
4200
4200
4201
4201
4202
4202
4203
4203
4204
4204
4205
4205
4206
4206
4207
4207
4208
4208
4209
4209
4210
4210
4211
4211
4212
4212
4213
4213
4214
4214
4215
4215
4216
4216
4217
4217
4218
4218
4219
4219
4220
4220
4221
4221
4222
4222
4223
4223
4224
4224
4225
4225
4226
4226
4227
4227
4228
4228
4229
4229
4230
4230
4231
4231
4232
4232
4233
4233
4234
4234
4235
4235
4236
4236
4237
4237
4238
4238
4239
4239
4240
4240
4241
4241
4242
4242
4243
4243
4244
4244
4245
4245
4246
4246
4247
4247
4248
4248
4249
4249
4250
4250
4251
4251
4252
4252
4253
4253
4254
4254
4255
4255
4256
4256
4257
4257
4258
4258
4259
4259
4260
4260
4261
4261
4262
4262
4263
4263
4264
4264
4265
4265
4266
4266
4267
4267
4268
4268
4269
4269
4270
4270
4271
4271
4272
4272
4273
4273
4274
4274
4275
4275
4276
4276
4277
4277
4278
4278
4279
4279
4280
4280
4281
4281
4282
4282
4283
4283
4284
4284
4285
4285
4286
4286
4287
4287
4288
4288
4289
4289
4290
4290
4291
4291
4292
4292
4293
4293
4294
4294
4295
4295
4296
4296
4297
4297
4298
4298
4299
4299
4300
4300
4301
4301
4302
4302
4303
4303
4304
4304
4305
4305
4306
4306
4307
4307
4308
4308
4309
4309
4310
4310
4311
4311
4312
4312
4313
4313
4314
4314
4315
4315
4316
4316
4317
4317
4318
4318
4319
4319
4320
4320
4321
4321
4322
4322
4323
4323
4324
4324
4325
4325
4326
4326
4327
4327
4328
4328
4329
4329
4330
4330
4331
4331
4332
4332
4333
4333
4334
4334
4335
4335
4336
4336
4337
4337
4338
4338
4339
4339
4340
4340
4341
4341
4342
4342
4343
4343
4344
4344
4345
4345
4346
4346
4347
4347
4348
4348
4349
4349
4350
4350
4351
4351
4352
4352
4353
4353
4354
4354
4355
4355
4356
4356
4357
4357
4358
4358
4359
4359
4360
4360
4361
4361
4362
4362
4363
4363
4364
4364
4365
4365
4366
4366
4367
4367
4368
4368
4369
4369
4370
4370
4371
4371
4372
4372
4373
4373
4374
4374
4375
4375
4376
4376
4377
4377
4378
4378
4379
4379
4380
4380
4381
4381
4382
4382
4383
4383
4384
4384
4385
4385
4386
4386
4387
4387
4388
4388
4389
4389
4390
4390
4391
4391
4392
4392
4393
4393
4394
4394
4395
4395
4396
4396
4397
4397
4398
4398
4399
4399
4400
4400
4401
4401
4402
4402
4403
4403
4404
4404
4405
4405
4406
4406
4407
4407
4408
4408
4409
4409
4410
4410
4411
4411
4412
4412
4413
4413
4414
4414
4415
4415
4416
4416
4417
4417
4418
4418
4419
4419
4420
4420
4421
4421
4422
4422
4423
4423
4424
4424
4425
4425
4426
4426
4427
4427
4428
4428
4429
4429
4430
4430
4431
4431
4432
4432
4433
4433
4434
4434
4435
4435
4436
4436
4437
4437
4438
4438
4439
4439
4440
4440
4441
4441
4442
4442
4443
4443
4444
4444
4445
4445
4446
4446
4447
4447
4448
4448
4449
4449
4450
4450
4451
4451
4452
4452
4453
4453
4454
4454
4455
4455
4456
4456
4457
4457
4458
4458
4459
4459
4460
4460
4461

```

```

28081415     loader, 'input', data, test_directory, host, port, None, None,
28091416         prefix=prefix,
28101417         handlers=handler_objects)
28111418 >>>>> REPLACE
28121419     '''
28131420     '''python
28141421     ### gabbi/runner.py
28151422 <<<<< SEARCH
28161423         success = run_suite(fh, handler_objects, host, port,
28171424                         prefix, force_ssl, failfast)
28181425 =====
28191426         test_directory = os.path.dirname(input_file) or '.'
28201427         success = run_suite(fh, handler_objects, host, port,
28211428                         prefix, force_ssl, failfast,
28221429                         test_directory=test_directory)
28231430 >>>>> REPLACE
28241431     '''
28251432     Answer 3:
28261433     '''python
28271434     ### gabbi/runner.py
28281435 <<<<< SEARCH
28291436     def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
28301437         False,
28311438             failfast=False):
28321439         """Run the tests from the YAML in handle."""
28331440         data = utils.load_yaml(handle)
28341441         if force_ssl:
28351442             if 'defaults' in data:
28361443                 data['defaults']['ssl'] = True
28371444             else:
28381445                 data['defaults'] = {'ssl': True}

28391446         loader = unittest.defaultTestLoader
28401447         test_suite = suitemaker.test_suite_from_dict(
28411448             loader, 'input', data, '.', host, port, None, None, prefix=
28421449                 prefix,
28431450                 handlers=handler_objects)

28441451         result = ConciseTestRunner(
28451452             verbosity=2, failfast=failfast).run(test_suite)
28461453         return result.wasSuccessful()
28471454 =====
28481455     def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
28491456         False,
28501457             failfast=False, test_directory='.'):
28511458         """Run the tests from the YAML in handle."""
28521459         data = utils.load_yaml(handle)
28531460         if force_ssl:
28541461             if 'defaults' in data:
28551462                 data['defaults']['ssl'] = True
28561463             else:
28571464                 data['defaults'] = {'ssl': True}

28581465         loader = unittest.defaultTestLoader
28591466         test_suite = suitemaker.test_suite_from_dict(
28601467             loader, 'input', data, test_directory, host, port, None, None,
28611468             prefix=prefix, handlers=handler_objects)

28621469         result = ConciseTestRunner(
28631470             verbosity=2, failfast=failfast).run(test_suite)
28641471         return result.wasSuccessful()
28651472 >>>>> REPLACE
28661473     '''
28671474     '''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  >>>>> REPLACE
2886 1499  ``
2887 1500
2888 1501 Answer 4:
2889 1502  '''python
2890 1503  ##### gabbi/runner.py
2891 1504  <<<<< SEARCH
2892 1505  def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2893 1506  False,
2894 1507  failfast=False):
2895 1508  =====
2896 1509  def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
2897 1510  False,
2898 1511  failfast=False, test_directory='.'):
2899 1512  >>>>> REPLACE
2900 1513  ``
2901 1514  '''python
2902 1515  ##### gabbi/runner.py
2903 1516  <<<<< SEARCH
2904 1517  test_suite = suitemaker.test_suite_from_dict(
2905 1518  loader, 'input', data, '.', host, port, None, None, prefix=
2906 1519  prefix,
2907 1520  handlers=handler_objects)
2908 1521  =====
2909 1522  test_suite = suitemaker.test_suite_from_dict(
2910 1523  loader, 'input', data, test_directory, host, port, None, None,
2911 1524  prefix=prefix, handlers=handler_objects)
2912 1525  >>>>> REPLACE
2913 1526  ``
2914 1527  '''python
2915 1528  ##### gabbi/runner.py
2916 1529  <<<<< SEARCH
2917 1530  for input_file in input_files:
2918 1531      with open(input_file, 'r') as fh:
2919 1532          success = run_suite(fh, handler_objects, host, port,
2920 1533                          prefix, force_ssl, failfast)
2921 1534  =====
2922 1535  for input_file in input_files:
2923 1536      # Determine test directory based on input file's location
2924 1537      test_directory = os.path.dirname(os.path.abspath(input_file))
2925 1538      or '.'
2926 1539      with open(input_file, 'r') as fh:

```

```

29161534         success = run_suite(fh, handler_objects, host, port,
29171535                     prefix, force_ssl, failfast,
29181536                     test_directory=test_directory)
29191537>>>>> REPLACE
29201538```
29211539Answer 5:
29221540```python
29231541### gabbi/runner.py
29241543<<<<< SEARCH
29251544import sys
29261545import unittest
29271546=====
29281548import os
29291549import sys
29301550import unittest
29311551>>>>> REPLACE
29321552```
29331553```python
29341555<<<<< SEARCH
29351556def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
29361556    False,
29371557        failfast=False):
29381558=====
29391559def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
29401560    False,
29411561        failfast=False, test_directory='.'):
29421562>>>>> REPLACE
29431563```
29441565```python
29451566<<<<< SEARCH
29461567    test_suite = suitemaker.test_suite_from_dict(
29471568        loader, 'input', data, '.', host, port, None, None, prefix=
29481569            prefix,
29491570            handlers=handler_objects)
29501571=====
29511572>>>>> REPLACE
29521573```
29531574```python
29541576<<<<< SEARCH
29551577    for input_file in input_files:
29561578        with open(input_file, 'r') as fh:
29571579            success = run_suite(fh, handler_objects, host, port,
29581581                prefix, force_ssl, failfast)
29591582=====
29601583    for input_file in input_files:
29611584        test_directory = os.path.dirname(input_file) or '.'
29621585        with open(input_file, 'r') as fh:
29631587            success = run_suite(fh, handler_objects, host, port,
29641588                prefix, force_ssl, failfast,
29651589                test_directory=test_directory)
29661590>>>>> REPLACE
29671591```
29681592Answer 6:
29691593```python
29701594<<<<< SEARCH
29711595    file_suite = suitemaker.test_suite_from_dict(

```

```

2970 1596
2971 1597
2972 1598
2973
2974 1599
2975 1600
2976 1601
2977 1602
2978
2979 1604
2980 1605
2981 1606
2982 1607
2983 1608
2984 1609
2985 1610
2986 1611
2987
2988 1613
2989 1614
2990 1615
2991 1616
2992 1617
2993 1618
2994 1619
2995 1620
2996 1621
2997 1622
2998
2999 1624
3000 1625
3001 1626
3002 1628
3003 1629
3004 1630
3005 1631
3006 1632
3007 1634
3008 1635
3009 1636
3010 1637
3011 1638
3012
3013 1640
3014 1641
3015 1642
3016 1643
3017 1644
3018 1645
3019 1647
3020 1648
3021 1649
3022 1650
3023 1651
3024 1652
3025 1653

        loader, test_base_name, suite_dict, path, host, port,
        fixture_module, intercept, prefix=prefix,
        test_loader_name=test_loader_name, handlers=handler_objects
        ,
        inner_fixtures=inner_fixtures)

=====
        test_file_dir = os.path.dirname(test_file)
        file_suite = suitemaker.test_suite_from_dict(
            loader, test_base_name, suite_dict, test_file_dir, host,
            port,
            fixture_module, intercept, prefix=prefix,
            test_loader_name=test_loader_name, handlers=handler_objects
            ,
            inner_fixtures=inner_fixtures)

>>>>> REPLACE
```
```python
### gabbi/runner.py
<<<<< SEARCH
def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
    False,
    failfast=False):
=====
def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
    False,
    failfast=False, test_directory='.'):

>>>>> REPLACE
```
```python
### gabbi/runner.py
<<<<< SEARCH
    test_suite = suitemaker.test_suite_from_dict(
        loader, 'input', data, '.', host, port, None, None, prefix=
            prefix,
        handlers=handler_objects)

=====
    test_suite = suitemaker.test_suite_from_dict(
        loader, 'input', data, test_directory, host, port, None, None,
        prefix=prefix, handlers=handler_objects)

>>>>> REPLACE
```
```python
### gabbi/runner.py
<<<<< SEARCH
    success = run_suite(fh, handler_objects, host, port,
        prefix, force_ssl, failfast)

=====
    test_directory = os.path.dirname(input_file)
    success = run_suite(fh, handler_objects, host, port,
        prefix, force_ssl, failfast,
        test_directory)

>>>>> REPLACE
```
```
Answer 7:
```python
gabbi/runner.py
<<<<< SEARCH
import argparse
from importlib import import_module
import sys
import unittest
=====
import argparse
from importlib import import_module

```

```

30241654 import os
30251655 import sys
30261656 import unittest
30271657 >>>>> REPLACE
30281658 '''
30291659 '''python
30301660 ### gabbi/runner.py
30311661 <<<<< SEARCH
30321662 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
30331663 False,
30341664 failfast=False):
30351665 """Run the tests from the YAML in handle."""
30361666 data = utils.load_yaml(handle)
30371667 if force_ssl:
30381669 if 'defaults' in data:
30391670 data['defaults']['ssl'] = True
30401672 else:
30411673 data['defaults'] = {'ssl': True}
30421674
30431675 loader = unittest.defaultTestLoader
30441676 test_suite = suitemaker.test_suite_from_dict(
30451677 loader, 'input', data, '.', host, port, None, None, prefix=
30461678 prefix,
30471679 handlers=handler_objects)
30481680 =====
30491681 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
30501682 False,
30511683 failfast=False, test_directory='.'):
30521684 """Run the tests from the YAML in handle."""
30531685 data = utils.load_yaml(handle)
30541686 if force_ssl:
30551687 if 'defaults' in data:
30561688 data['defaults']['ssl'] = True
30571689 else:
30581691 data['defaults'] = {'ssl': True}
30591692
30601693 loader = unittest.defaultTestLoader
30611694 test_suite = suitemaker.test_suite_from_dict(
30621696 loader, 'input', data, test_directory, host, port, None, None,
30631697 prefix=prefix, handlers=handler_objects)
30641698 >>>>> REPLACE
30651699 '''
30661700 '''python
30671701 ### gabbi/runner.py
30681702 <<<<< SEARCH
30691703 with open(input_file, 'r') as fh:
30701704 success = run_suite(fh, handler_objects, host, port,
30711705 prefix, force_ssl, failfast)
30721707 if not failure: # once failed, this is considered
30731708 immutable
30741709 failure = not success
30751710 if failure and failfast:
30761711 break
30771711 =====
30781712 test_dir = os.path.dirname(os.path.abspath(input_file))
30791713 with open(input_file, 'r') as fh:
30801714 success = run_suite(fh, handler_objects, host, port,
30811715 prefix, force_ssl, failfast,
30821716 test_directory=test_dir)
30831717 if not failure: # once failed, this is considered
30841718 immutable
30851719 failure = not success
30861720 if failure and failfast:
30871721 break
30881722 >>>>> 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 1722 if 'defaults' in data:
3087 1723 data['defaults']['ssl'] = True
3088 1724 else:
3089 1725 data['defaults'] = {'ssl': True}
3090 1726
3091 1727 loader = unittest.defaultTestLoader
3092 1728 test_suite = suitemaker.test_suite_from_dict(
3093 1729 loader, 'input', data, '.', host, port, None, None, prefix=
3094 1730 prefix,
3095 1731 handlers=handler_objects)
3096 1732 =====
3097 1733 data = utils.load_yaml(handle)
3098 1734 if force_ssl:
3099 1735 if 'defaults' in data:
3100 1736 data['defaults']['ssl'] = True
3101 1737 else:
3102 1738 data['defaults'] = {'ssl': True}
3103 1739
3104 1740 # Determine test directory based on input handle
3105 1741 if hasattr(handle, 'name'):
3106 1742 test_directory = os.path.dirname(os.path.abspath(handle.name))
3107 1743 else:
3108 1744 test_directory = '..'
3109 1745
3110 1746 loader = unittest.defaultTestLoader
3111 1747 test_suite = suitemaker.test_suite_from_dict(
3112 1748 loader, 'input', data, test_directory, host, port, None, None,
3113 1749 prefix=prefix, handlers=handler_objects)
3114 1750 >>>>> REPLACE
3115 1751 ```
3116 1752 Answer 9:
3117 1753 ```python
3118 1754 ### gabbi/driver.py
3119 1755 <<<<< SEARCH
3120 1756 for test_file in glob.iglob('%s/*.yaml' % path):
3121 1757 if '_' in os.path.basename(test_file):
3122 1758 warnings.warn(exception.GabbiSyntaxWarning(
3123 1759 "'_' in test filename %s. This can break suite grouping
3124 1760 ."
3125 1761 % test_file))
3126 1762 if intercept:
3127 1763 host = str(uuid.uuid4())
3128 1764 suite_dict = utils.load_yaml(yaml_file=test_file)
3129 1765 test_base_name = os.path.splitext(os.path.basename(test_file))
3130 1766 [0]
3131 1767 if all_test_base_name:
3132 1768 test_base_name = '%s_%s' % (all_test_base_name,
3133 1769 test_base_name)
3134 1770
3135 1771 if require_ssl:
3136 1772 if 'defaults' in suite_dict:
3137 1773 suite_dict['defaults']['ssl'] = True
3138 1774 else:
3139 1775 suite_dict['defaults'] = {'ssl': True}
3140 1776
3141 1777 file_suite = suitemaker.test_suite_from_dict(
3142 1778

```

```

31321774 loader, test_base_name, suite_dict, path, host, port,
31331775 fixture_module, intercept, prefix=prefix,
31341776 test_loader_name=test_loader_name, handlers=handler_objects
3135
31361777 ,
31371778 inner_fixtures=inner_fixtures)
31381779 =====
31391780 for test_file in glob.iglob('%s/*.yaml' % path):
31401781 if '_' in os.path.basename(test_file):
31411782 warnings.warn(exception.GabbiSyntaxWarning(
31421783 "'_' in test filename %s. This can break suite grouping
31431784 ."
31441785 % test_file))
31451786 if intercept:
31461787 host = str(uuid.uuid4())
31471788 suite_dict = utils.load_yaml(yaml_file=test_file)
31481789 test_base_name = os.path.splitext(os.path.basename(test_file))
31491790 [0]
31501791 if all_test_base_name:
31511792 test_base_name = '%s_%s' % (all_test_base_name,
31521793 test_base_name)
31531794
31541795 if require_ssl:
31551797 if 'defaults' in suite_dict:
31561798 suite_dict['defaults']['ssl'] = True
31571799 else:
31581800 suite_dict['defaults'] = {'ssl': True}
31591801
31601802 # Use absolute path of the YAML file's directory
31611803 test_file_dir = os.path.dirname(os.path.abspath(test_file))
31621804 file_suite = suitemaker.test_suite_from_dict(
31631805 loader, test_base_name, suite_dict, test_file_dir, host,
31641806 port,
31651807 fixture_module, intercept, prefix=prefix,
31661808 test_loader_name=test_loader_name, handlers=handler_objects
31671809
31681810 ,
31691811 inner_fixtures=inner_fixtures)
31701812 >>>>> REPLACE
31711813 ``
31721814
Answer 10:
31731815 ```python
31741816 ### gabbi/runner.py
31751817 <<<<< SEARCH
31761818 import argparse
31771819 from importlib import import_module
31781820 import sys
31791821 import unittest
31801822 =====
31811823 import argparse
31821824 from importlib import import_module
31831825 import os
31841826 import sys
31851827 import unittest
31861828 >>>>> REPLACE
31871829 ``
31881830 ```python
31891831 ### gabbi/runner.py
31901832 <<<<< SEARCH
31911833 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
31921834 False,
31931835 failfast=False):
31941836
31951837 =====
31961838 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
31971839 False,
31981840 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 =====
3197 1842 test_suite = suitemaker.test_suite_from_dict(
3198 1843 loader, 'input', data, test_directory, host, port, None, None,
3199 1844 prefix=prefix, force_ssl=force_ssl, failfast=failfast,
3200 1845 handlers=handler_objects)
3201 1846 >>>>> REPLACE
3202 1847 ``
3203 1848 '''python
3204 1849 ### gabbi/runner.py
3205 1850 <<<<< SEARCH
3206 1851 success = run_suite(fh, handler_objects, host, port,
3207 1852 prefix, force_ssl, failfast)
3208 1853 =====
3209 1854 test_dir = os.path.dirname(input_file)
3210 1855 success = run_suite(fh, handler_objects, host, port,
3211 1856 prefix, force_ssl, failfast,
3212 1857 test_directory=test_dir)
3213 1858 >>>>> REPLACE
3214 1859 ``
3215 1860 '''python
3216 1861 ### gabbi/case.py
3217 1862 <<<<< SEARCH
3218 1863 path = os.path.join(self.test_directory, os.path.basename(filename)
3219 1864)
3220 1865 =====
3221 1866 path = os.path.join(self.test_directory, filename)
3222 1867 >>>>> REPLACE
3223 1868 ``
Answer 11:
3224 1869 '''python
3225 1870 ### gabbi/runner.py
3226 1871 <<<<< SEARCH
3227 1872 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3228 1873 False,
3229 1874 failfast=False):
3230 1875 """Run the tests from the YAML in handle."""
3231 1876 data = utils.load_yaml(handle)
3232 1877 if force_ssl:
3233 1878 if 'defaults' in data:
3234 1879 data['defaults']['ssl'] = True
3235 1880 else:
3236 1881 data['defaults'] = {'ssl': True}
3237 1882
3238 1883 loader = unittest.defaultTestLoader
3239 1884 test_suite = suitemaker.test_suite_from_dict(
3240 1885 loader, 'input', data, '.', host, port, None, None, prefix=
3241 1886 prefix,
3242 1887 handlers=handler_objects)
3243 1888 =====
3244 1889 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3245 1890 False,
3246 1891 failfast=False, test_directory='.'):
3247 1892 """Run the tests from the YAML in handle."""
3248 1893 data = utils.load_yaml(handle)

```

```

32401891 if force_ssl:
32411892 if 'defaults' in data:
32421893 data['defaults']['ssl'] = True
32431894 else:
32441895 data['defaults'] = {'ssl': True}
32451896
32461897 loader = unittest.defaultTestLoader
32471898 test_suite = suitemaker.test_suite_from_dict(
32481899 loader, 'input', data, test_directory, host, port,
32491900 None, None, prefix=prefix, handlers=handler_objects)
32501901 >>>>> REPLACE
32511902 ``
32521903 '''python
32531904 ### gabbi/runner.py
32541905 <<<<< SEARCH
32551906 success = run_suite(fh, handler_objects, host, port,
32561907 prefix, force_ssl, failfast)
32571908 =====
32581909 test_dir = os.path.dirname(input_file)
32591910 success = run_suite(fh, handler_objects, host, port,
32601911 prefix, force_ssl, failfast,
32611912 test_directory=test_dir)
32621913 >>>>> REPLACE
32631914 ``
32641915
Answer 12:
32651916 '''python
32661917 ### gabbi/runner.py
32671918 <<<<< SEARCH
32681919 import argparse
32691920 from importlib import import_module
32701921 import sys
32711922 import unittest
32721923 =====
32731924 import argparse
32741925 from importlib import import_module
32751926 import os
32761927 import sys
32771928 import unittest
32781929 >>>>> REPLACE
32791930 ``
32801931 '''python
32811932 ### gabbi/runner.py
32821933 <<<<< SEARCH
32831934 else:
32841935 for input_file in input_files:
32851936 with open(input_file, 'r') as fh:
32861937 success = run_suite(fh, handler_objects, host, port,
32871938 prefix, force_ssl, failfast)
32881939 =====
32891940 else:
32901941 for input_file in input_files:
32911942 test_dir = os.path.dirname(os.path.abspath(input_file))
32921943 with open(input_file, 'r') as fh:
32931944 success = run_suite(fh, handler_objects, host, port,
32941945 prefix, force_ssl, failfast,
32951946 test_directory=test_dir)
32961947 >>>>> REPLACE
32971948 ``
32981949 '''python
32991950 ### gabbi/runner.py
33001951 <<<<< SEARCH
33011952 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
33021953 False,
33031954 failfast=False):

```

```

32941955 """Run the tests from the YAML in handle."""
32951956 data = utils.load_yaml(handle)
32961957 if force_ssl:
32971958 if 'defaults' in data:
32981959 data['defaults']['ssl'] = True
32991960 else:
33001961 data['defaults'] = {'ssl': True}
33011962
33021963 loader = unittest.defaultTestLoader
33031964 test_suite = suitemaker.test_suite_from_dict(
33041965 loader, 'input', data, '.', host, port, None, None, prefix=
33051966 prefix,
33061967 handlers=handler_objects)
33071968 =====
33081969 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
33091970 False,
33101971 failfast=False, test_directory='.'):
33111972 """Run the tests from the YAML in handle."""
33121973 data = utils.load_yaml(handle)
33131974 if force_ssl:
33141975 if 'defaults' in data:
33151976 data['defaults']['ssl'] = True
33161977 else:
33171978 data['defaults'] = {'ssl': True}
33181979
33191980 loader = unittest.defaultTestLoader
33201981 test_suite = suitemaker.test_suite_from_dict(
33211982 loader, 'input', data, test_directory, host, port, None, None,
33221983 prefix=prefix, handlers=handler_objects)
33231984 >>>>> REPLACE
33241985 ``
33251986
33261987 Answer 13:
33271988 ``
33281989 """python
33291990 gabbi/runner.py
33301991 <<<<< SEARCH
33311992 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
33321993 False,
33331994 failfast=False):
33341995 """Run the tests from the YAML in handle."""
33351996 data = utils.load_yaml(handle)
33361997 if force_ssl:
33371998 if 'defaults' in data:
33381999 data['defaults']['ssl'] = True
33392000 else:
33402001 data['defaults'] = {'ssl': True}
33412002
33422003 loader = unittest.defaultTestLoader
33432004 test_suite = suitemaker.test_suite_from_dict(
33442005 loader, 'input', data, '.', host, port, None, None, prefix=
33452006 prefix,
33462007 handlers=handler_objects)
33472008 =====
33482009 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
33492010 False,
33502011 failfast=False, test_directory='.'):
33512012 """Run the tests from the YAML in handle."""
33522013 data = utils.load_yaml(handle)
33532014 if force_ssl:
33542015 if 'defaults' in data:
33552016 data['defaults']['ssl'] = True
33562017 else:
33572018 data['defaults'] = {'ssl': True}
33582019
33592020 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 >>>>> REPLACE
3368 2035 '''
3369 2036
3370 2037 Answer 14:
3371 2038 '''python
3372 2039 ### gabbi/runner.py
3373 2040 <<<<< SEARCH
3374 2041 for input_file in input_files:
3375 2042 with open(input_file, 'r') as fh:
3376 2043 success = run_suite(fh, handler_objects, host, port,
3377 2044 prefix, force_ssl, failfast)
3378 2045 =====
3379 2046 for input_file in input_files:
3380 2047 with open(input_file, 'r') as fh:
3381 2048 success = run_suite(input_file, fh, handler_objects,
3382 2049 host, port,
3383 2050 prefix, force_ssl, failfast)
3384 2051 >>>>> REPLACE
3385 2052 '''
3386 2053 '''python
3387 2054 ### gabbi/runner.py
3388 2055 <<<<< SEARCH
3389 2056 def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
3390 2057 False,
3391 2058 failfast=False):
3392 2059 """Run the tests from the YAML in handle."""
3393 2060 data = utils.load_yaml(handle)
3394 2061 if force_ssl:
3395 2062 if 'defaults' in data:
3396 2063 data['defaults']['ssl'] = True
3397 2064 else:
3398 2065 data['defaults'] = {'ssl': True}
3399 2066
3400 2067 loader = unittest.defaultTestLoader
3401 2068 test_directory = '.'
3402 2069 test_suite = suitemaker.test_suite_from_dict(
3403 2070 loader, 'input', data, test_directory, host, port, None, None,
3404 2071 prefix=prefix,
3405 2072 handlers=handler_objects)
3406 2073 =====
3407 2074 def run_suite(input_file, handle, handler_objects, host, port, prefix,
3408 2075 force_ssl=False,
3409 2076 failfast=False):
3410 2077 """Run the tests from the YAML in handle."""
3411 2078 data = utils.load_yaml(handle)
3412 2079 if force_ssl:

```

```

34022076 if 'defaults' in data:
34032077 data['defaults']['ssl'] = True
34042078 else:
34052079 data['defaults'] = {'ssl': True}
34062080
34072081 loader = unittest.defaultTestLoader
34082082 test_directory = os.path.dirname(input_file) if input_file else '.'
34092083 test_suite = suitemaker.test_suite_from_dict(
34102084 loader, 'input', data, test_directory, host, port, None, None,
34112085 prefix=prefix,
34122086 handlers=handler_objects)
34132087 >>>>> REPLACE
34142088 ``
34152089 ````python
34162090 ##### gabbi/runner.py
34172091 <<<<< SEARCH
34182092 if not input_files:
34192093 success = run_suite(sys.stdin, handler_objects, host, port,
34202094 prefix, force_ssl, failfast)
34212095 =====
34222096 if not input_files:
34232097 success = run_suite(None, sys.stdin, handler_objects, host,
34242098 port,
34252099 prefix, force_ssl, failfast)
34262100 >>>>> REPLACE
34272101 ``
34282102 ```` END FILE ---
34292103
34302104 Please first summary and analyze the key differences between the
34312105 trajectories, and then generate *SEARCH/REPLACE* edits to fix the
34322106 issue.
34332107
34342108 Every *SEARCH/REPLACE* edit must use this format:
34352109 1. The file path
34362110 2. The start of search block: <<<<< SEARCH
34372111 3. A contiguous chunk of lines to search for in the existing source
34382112 code
34392113 4. The dividing line: =====
34402114 5. The lines to replace into the source code
34412115 6. The end of the replace block: >>>>> REPLACE
34422116
34432117 Here is an example:
34442118
34452119 ````python
34462120 ##### mathweb/flask/app.py
34472121 <<<<< SEARCH
34482122 from flask import Flask
34492123 =====
34502124 import math
34512125 from flask import Flask
34522126 >>>>> REPLACE
34532127 ``
34542128 Please note that the *SEARCH/REPLACE* edit REQUIRES PROPER INDENTATION.
34552129 If you would like to add the line ' print(x)', you must
34562130 fully write that out, with all those spaces before the code!
34572131 Wrap the *SEARCH/REPLACE* edit in blocks ````python...````.
34582132 The summary of the key differences between the trajectories should be
34592133 in the thinking part.
34602134
34612135 Response:
34622136
34632137 <think>

```

3456 2133 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.  
 3462 2134 First, I need to understand how the current code is handling the data  
 3463 file paths. The user mentioned that in `_load_data_file`, the code  
 3464 uses `os.path.join(self.test_directory, os.path.basename(filename))`.  
 3465 This takes the `test_directory` (which defaults to '.'), discards any  
 3466 directories in the filename, and appends the basename. That's why if  
 3467 the YAML is in `/my/tests/test.yaml` and the data is <@data.txt, it  
 3468 2135 looks for `./data.txt` instead of `/my/tests/data.txt`.  
 3469 2137 The key here is to make sure that the `test_directory` is set to the  
 3470 directory where the YAML file is located. Then, when joining with  
 3471 the filename (without basename), it would correctly resolve the  
 3472 2138 relative path.  
 3473 2139 Looking at the provided code snippets, the main changes are in `runner.  
 3474 py` where the `test_directory` is determined. The user's proposed  
 3475 solutions involve modifying the `run_suite` function to accept a  
 3476 `test_directory` parameter, which is derived from the input file's  
 3477 2140 directory.  
 3478 2141 In the original code, when running a suite from an input file, the  
 3479 `test_directory` is hardcoded to '.'. The various answers modify this  
 3480 by passing the directory of the input file as `test_directory`. For  
 3481 example, in Answer 10, they add the `test_directory` parameter to  
 3482 `run_suite` and set it to `os.path.dirname(input_file)`. Then, in `case.  
 3483 py`, they change the path joining to use the filename directly  
 3484 2142 instead of basename.  
 3485 2143 Wait, the critical part is in `case.py`'s `_load_data_file` method. The  
 3486 original code uses `os.path.basename(filename)`, which strips any  
 3487 directories from the filename. So if the YAML has <@test\_data/file.  
 3488 txt, the basename is `file.txt`, and it's joined with `test_directory`.  
 3489 But if `test_directory` is the YAML's directory, then it should be  
 3490 `YAML_DIR/test_data/file.txt`. However, the current code would do  
 3491 `test_directory + file.txt`, which would be `YAML_DIR/file.txt`. That's  
 3492 2144 incorrect. So the problem is that the code is taking the basename,  
 3493 2145 thus ignoring any subdirectories in the filename.  
 3494 Ah, so the fix here is to not take the basename. Instead, the filename  
 3495 after <@ should be treated as a relative path from the YAML's  
 3496 directory. So in `case.py`'s `_load_data_file`, instead of using `os.path  
 3497 .basename(filename)`, we should use the filename as-is. Then, `os.path  
 3498 2146 .join(self.test_directory, filename)` would correctly resolve the  
 3499 2147 path relative to the YAML's directory.  
 3500 But wait, the filename could be an absolute path. For example, if the  
 3501 YAML has <@/absolute/path/file.txt, then `os.path.join` would just use  
 3502 the absolute path. However, the documentation says that the  
 3503 2148 filename is relative to the YAML file. So absolute paths might not  
 3504 2149 be intended. But the code should handle that.  
 3505 So combining two changes: in `runner.py`, set `test_directory` to the YAML  
 3506 file's directory, and in `case.py`, stop using basename and use the  
 3507 2150 full filename.  
 3508 Looking at Answer 10, there's a change in `case.py` from `os.path.basename  
 3509 (filename)` to `filename`. That's exactly the fix needed. Then, in  
 3510 `runner.py`, when running the suite, pass the `test_directory` as the  
 3511 directory of the input file.

3510<sup>2152</sup>  
 3511<sup>2153</sup> So the key files to modify are:  
 3512<sup>2154</sup>  
 3513<sup>2155</sup> 1. gabbi/runner.py: When calling `run_suite`, pass `test_directory` as the  
 3514 directory of the input file. This involves modifying the `run_suite`  
 3515 function to accept `test_directory` as a parameter and then passing it  
 3516 when the input file is processed.  
 3517<sup>2157</sup>  
 3518 2. gabbi/case.py: Change the path construction from using `os.path.`  
 3519 `basename(filename)` to using the filename directly. This allows  
 3520<sup>2158</sup> subdirectories in the filename to be included in the path.  
 3521<sup>2159</sup>  
 3522<sup>2160</sup> Looking through the provided code snippets, the one that does both is  
 3523<sup>2161</sup> Answer 10. Let me check:  
 3524  
 3525  
 3526  
 3527<sup>2162</sup>  
 3528<sup>2163</sup> In Answer 10, the `runner.py` changes add the `test_directory` parameter  
 3529 and set it to `os.path.dirname(input_file)`. Then, in `case.py`, the  
 3530 path is changed to `os.path.join(self.test_directory, filename)`. That  
 3531 including any subdirectories in the filename.  
 3532<sup>2165</sup>  
 3533<sup>2166</sup> But wait, the original code in `case.py` uses `os.path.basename(filename)`.  
 3534<sup>2167</sup> So changing that to filename means that if the input is '`test_data/`  
 3535 `file.txt`', it's joined as `test_directory/test_data/file.txt`. If  
 3536 `test_directory` is the YAML's directory, that's correct.  
 3537  
 3538 ----- Omit some reasoning content -----  
 3539  
 3540 But perhaps better to use `os.path.abspath` and then `dirname` to handle  
 3541 relative paths correctly. But maybe the existing code already works.  
 3542 For example, if the `input_file` is '`tests/test.yaml`', then `os.path.`  
 3543 `dirname` would return '`tests`', which when joined with `os.getcwd()`  
 3544 would be correct. Wait, no. `os.path.dirname` just gives the directory  
 3545 part of the path. So if `input_file` is a relative path, like '`tests/`  
 3546 `test.yaml`', then `test_directory` would be '`tests`', and when `os.path.`  
 3547 `join` is called, it's relative to the current working directory. But  
 3548 the code in `suitemaker.test_suite_from_dict` might expect an absolute  
 3549 path. However, the original code uses '`..`' as the `test_directory`,  
 3550 which is relative. So perhaps using the directory of the input file  
 3551 as `test_directory` is sufficient.  
 3552<sup>2168</sup>  
 3553<sup>2170</sup>  
 3554<sup>2171</sup> Another thing to consider: when the YAML is loaded from `stdin` (like via  
 3555<sup>2172</sup> redirect), the `test_directory` remains '`..`'. That's acceptable, as  
 3556<sup>2173</sup> the documentation mentions that when using `<`, the filename isn't  
 3557<sup>2174</sup> seen by Gabbi, so it's not possible to resolve the path relative to  
 3558<sup>2175</sup> the YAML's original location. So in that case, the `test_directory`  
 3559<sup>2176</sup> defaults to '`..`'.  
 3560<sup>2177</sup>  
 3561<sup>2178</sup> Now, in `case.py`:  
 3562<sup>2179</sup>  
 3563<sup>2180</sup> Original line:  
 3564<sup>2181</sup>  
 3565<sup>2182</sup> `path = os.path.join(self.test_directory, os.path.basename(filename))`  
 3566<sup>2183</sup>  
 3567<sup>2184</sup> Modified line:  
 3568<sup>2185</sup>  
 3569<sup>2186</sup> `path = os.path.join(self.test_directory, filename)`  
 3570<sup>2187</sup>  
 3571<sup>2188</sup> So the SEARCH/REPLACE block for `case.py`:  
 3572<sup>2189</sup>  
 3573<sup>2190</sup> `### gabbi/case.py`  
 3574<sup>2191</sup> `<<<<< SEARCH`  
 3575<sup>2192</sup>  `path = os.path.join(self.test_directory, os.path.basename(filename))`  
 3576<sup>2193</sup>  
 3577<sup>2194</sup> `=====`  
 3578<sup>2195</sup>

```

35642187 path = os.path.join(self.test_directory, filename)
35652188 >>>>> REPLACE
35662189
35672190 Putting all these together, the final edits are:
35682191
35692192
35702193
35712194
35722195
35732196
35742197
35752198
35762199
35772200 So the final answer would include these four SEARCH/REPLACE blocks.
35782201 </think>
35792202
35802203
35812204
35822205
35832206
35842207
35852208
35862209
35872210
35882211
35892212
35902213
35912214
35922215
35932216
35942217
35952218
35962219
35972220
35982221
35992222
36002223
36012224
36022225
36032226
36042227
36052228
36062229
36072230
36082231
36092232
36102233
36112234
36122235
36132236
36142237
36152238
36162239
36172240
36182241
36192242
36202243
36212244
36222245
36232246
    ````python
    ### gabbi/runner.py
    <<<<< SEARCH
    import argparse
    from importlib import import_module
    import sys
    import unittest
    =====
    import argparse
    from importlib import import_module
    import os
    import sys
    import unittest
    >>>>> REPLACE
    ````

    ````python
    ### gabbi/runner.py
    <<<<< SEARCH
    def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
        False,
                    failfast=False):
    =====
    def run_suite(handle, handler_objects, host, port, prefix, force_ssl=
        False,
                    failfast=False, test_directory='.'):
    >>>>> REPLACE
    ````

    ````python
    ### gabbi/runner.py
    <<<<< SEARCH
    test_suite = suitemaker.test_suite_from_dict(
        loader, 'input', data, '.', host, port, None, None, prefix=
            prefix,
        handlers=handler_objects)
    =====
        test_suite = suitemaker.test_suite_from_dict(
            loader, 'input', data, test_directory, host, port, None, None,
            prefix=prefix, handlers=handler_objects)
    >>>>> REPLACE
    ````

    ````python
    ### gabbi/runner.py
    <<<<< SEARCH
        else:

```

```

36182247     for input_file in input_files:
36192248         with open(input_file, 'r') as fh:
36202249             success = run_suite(fh, handler_objects, host, port,
36212250                             prefix, force_ssl, failfast)
36222251 =====
36232252     else:
36242253         for input_file in input_files:
36252254             test_directory = os.path.dirname(input_file) or '.'
36262255             with open(input_file, 'r') as fh:
36272256                 success = run_suite(fh, handler_objects, host, port,
36282257                               prefix, force_ssl, failfast,
36292258                               test_directory=test_directory)
36302259 >>>>> REPLACE
36312260 ``
36322261 ````python
36332262 ##### gabbi/case.py
36342263 <<<<< SEARCH
36352264     path = os.path.join(self.test_directory, os.path.basename(filename)
36362265         )
36372266 =====
36382267     path = os.path.join(self.test_directory, filename)
36392268 >>>>> REPLACE
36402269 ``
36412270 These changes:
36422271 1. Add 'os' import needed for path handling
36432272 2. Modify run_suite to accept a test_directory parameter
36442273 3. Use the input file's directory as the test_directory when processing
36452274     files
36462275 4. Update data file path resolution to preserve directory structure
36472276     from YAML relative paths
36482277 5. Remove basename handling that was stripping directory information
36492278     from data file paths
36502279
36512280 The net effect is that data files specified with '<@filename' in YAML
36522281 tests will now be resolved relative to their YAML file's directory,
36532282 supporting both flat and nested directory structures for test data.
36542283 When tests are run from stdin (using '<'), the default directory
36552284 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.