Answers to Reviewer Questions

ACM Reference Format:
. 2025. Answers to Reviewer Questions. 1, 1 (November 2025), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

0.1 Answer to RAQ6: Try on LiveCodeBench

Motivation&Approach. Per the reviewer's suggestion, we apply FDAT to the LiveCodeBench dataset. Because each task in LiveCodeBench provides a single complex algorithmic problem with a multi-line input string, it is difficult to locate subfunctions whose output types naturally match this representation. To address this, during type matching we treat the input string as a list (each line treated as one list element) and search for compatible subfunctions under this representation.

Using this strategy, we identify 81 tasks for which suitable subfunctions can be found via type matching, and evaluate them using GPT-5. For comparison, we include the strongest baselines, PPM-T and PPM-V.

Results. As shown in Table 1, we report pass@k for all methods. Because LiveCodeBench tasks have a single string input, only the SID variant of FDAT is applicable. The results show that SID achieves consistently lower pass@k than both PPM-T and PPM-V, indicating that FDAT constructs more challenging tasks LLMs.

Table 1. Comparison of FDAT (SID) and PPM Baselines on LiveCodeBench (pass@k)

	pass@1	pass@3	pass@5	pass@7	pass@10
PPM-T	0.17	0.29	0.35	0.38	0.42
PPM-V	0.17	0.30	0.37	0.41	0.44
SID	0.04	0.11	0.16	0.20	0.25

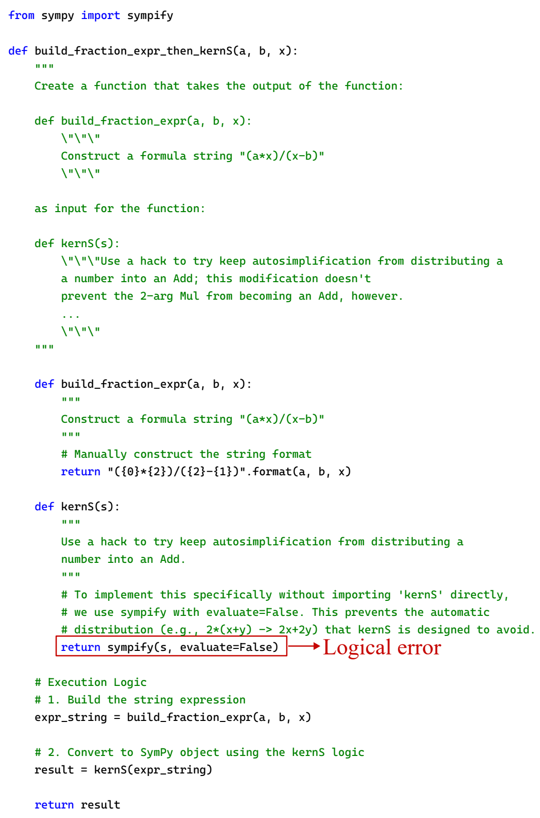
0.2 Answer to RBQ2: Comparison with SWE-bench

Motivation&Approach. To clarify the novelty of FDAT and address the reviewer's concern, we present a concrete case study comparing FDAT with SWE-bench. Specifically, we select the best-performing model reported on SWE-bench—Gemini 3 Pro Preview—and choose one SWE-bench instance that this model successfully fixes (sympy-19637, `kernS`). We then apply FDAT to construct a new task based on the same target function.

Results. The result reveals a clear gap between the two evaluation paradigms. As shown in Figure 1, we replace the input of the original function `kernS` with a simple helper function `build_fraction_expr`, thereby forming a new composed task under FDAT. Although Gemini

Author's Contact Information:

ACM XXXX-XXXX/2025/11-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn



```python
from sympy import sympify

def build_fraction_expr_then_kernS(a, b, x):
    """
    Create a function that takes the output of the function:

    def build_fraction_expr(a, b, x):
        \"\"\"
        Construct a formula string "(a*x)/(x-b)"
        \"\"\"

    as input for the function:

    def kernS(s):
        \"\"\"Use a hack to try keep autosimplification from distributing a
        a number into an Add; this modification doesn't
        prevent the 2-arg Mul from becoming an Add, however.
        ...
        \"\"\"
    """

    def build_fraction_expr(a, b, x):
        """
        Construct a formula string "(a*x)/(x-b)"
        """
        # Manually construct the string format
        return "({0}*{2})/({2}-{1})".format(a, b, x)

    def kernS(s):
        """
        Use a hack to try keep autosimplification from distributing a
        number into an Add.
        """
        # To implement this specifically without importing 'kernS' directly,
        # we use sympify with evaluate=False. This prevents the automatic
        # distribution (e.g., 2*(x+y) -> 2x+2y) that kernS is designed to avoid.
        return sympify(s, evaluate=False)

    # Execution Logic
    # 1. Build the string expression
    expr_string = build_fraction_expr(a, b, x)

    # 2. Convert to SymPy object using the kernS logic
    result = kernS(expr_string)

    return result
```

Fig. 1. Example of FDAT-Composed Task That Gemini 3 Pro Preview Fails to Solve

can correctly repair the real-world SWE-bench bug, it fails to generate correct code for the FDAT-composed task. The generated implementation incorrectly treats `sympify(s, evaluate=False)` as equivalent to `kernS(s)`, while the two behave differently for inputs containing unary minus, demonstrating a real error.

This example highlights a core novelty of FDAT: functional composition introduces new semantic dependencies that do not exist in the original tasks, thereby exposing reasoning and modular-integration weaknesses that benchmarks like SWE-bench cannot reveal—even when the model succeeds on the real-world issue itself. Rather than approximating SWE-bench, FDAT complements it by systematically generating new, compositional tasks that probe a different and currently under-evaluated ability: robust functional reasoning under decomposition.

0.3 Answer to RCQ1: Outdated LLMs

Motivation&Approach. Following the reviewer's suggestion, we evaluate one of the latest models, GPT-5, using HumanEval as the base benchmark. Due to time constraints, we compare FDAT with the strongest baselines, PPM-T and PPM-V.

Results. As shown in Table 2, (1) for the most critical metric, pass@1, FDAT's SID is slightly higher than PPM-V in one case, but in all other cases, FDAT is lower than both PPM-T and PPM-V; (2) for AID, pass@k is lower than both PPM-T and PPM-V across all settings, indicating that it poses the greatest challenge for current models. These results are consistent with the findings in RQ2, showing that our method provides a stricter evaluation, reducing pass@k—especially pass@1—and that AID significantly outperforms the baselines.

Table 2. GPT-5 Performance (pass@k) on HumanEval

	pass@1	pass@3	pass@5	pass@7	pass@10
PPM-T	0.61	0.65	0.66	0.66	0.66
PPM-V	0.51	0.58	0.60	0.61	0.61
AID	0.17	0.29	0.34	0.37	0.41
MRD	0.47	0.59	0.63	0.65	0.67
SID	0.52	0.69	0.75	0.79	0.82

0.4 Answer to RCQ2: Results should be separated

Results. The specific value results of Pass@k are shown in Tables 3 and Table 4.Note that these results correspond exactly to the data visualized in Figure 6 of the main paper.

Table 3. The Effectiveness Evaluation on the HumanEval Dataset

Methods	Incoder-1B					CodeGen-2B				
	pass@1	pass@3	pass@5	pass@7	pass@10	pass@1	pass@3	pass@5	pass@7	pass@10
Base	0.06	0.11	0.13	0.14	0.16	0.10	0.17	0.21	0.23	0.25
Insert_line	0.05	0.09	0.12	0.13	0.16	0.11	0.19	0.23	0.26	0.29
Comment	0.03	0.07	0.09	0.11	0.12	0.09	0.15	0.19	0.21	0.23
PPM-T	0.01	0.02	0.02	0.02	0.02	0.01	0.02	0.03	0.03	0.04
PPM-V	0.01	0.01	0.02	0.02	0.03	0.01	0.02	0.03	0.03	0.03
AID	0.01	0.02	0.03	0.04	0.06	0.00	0.00	0.00	0.00	0.00
MRD	0.00	0.01	0.02	0.03	0.04	0.02	0.06	0.09	0.13	0.19
SID	0.00	0.01	0.02	0.03	0.05	0.01	0.03	0.05	0.07	0.09
Methods	CodeGen2-1B					Santacoder				
	pass@1	pass@3	pass@5	pass@7	pass@10	pass@1	pass@3	pass@5	pass@7	pass@10
Base	0.06	0.10	0.12	0.13	0.14	0.15	0.22	0.26	0.28	0.30
Insert_line	0.07	0.10	0.11	0.12	0.13	0.13	0.19	0.22	0.24	0.25
Comment	0.05	0.09	0.10	0.11	0.11	0.11	0.18	0.21	0.23	0.25
PPM-T	0.01	0.02	0.03	0.03	0.04	0.02	0.04	0.05	0.06	0.06
PPM-V	0.00	0.01	0.01	0.02	0.02	0.01	0.02	0.03	0.04	0.04
AID	0.00	0.00	0.00	0.00	0.00	0.01	0.04	0.06	0.08	0.12
MRD	0.00	0.01	0.02	0.03	0.04	0.01	0.04	0.07	0.09	0.11
SID	0.00	0.01	0.02	0.02	0.03	0.03	0.07	0.10	0.13	0.16
Methods	Llama3-1b					Chatgpt				
	pass@1	pass@3	pass@5	pass@7	pass@10	pass@1	pass@3	pass@5	pass@7	pass@10
Base	0.14	0.22	0.26	0.29	0.32	0.59	0.78	0.83	0.86	0.87
Insert_line	0.14	0.23	0.28	0.31	0.35	0.63	0.79	0.83	0.84	0.86
Comment	0.12	0.19	0.23	0.25	0.27	0.75	0.84	0.87	0.88	0.89
PPM-T	0.03	0.05	0.06	0.07	0.08	0.26	0.41	0.46	0.49	0.52
PPM-V	0.01	0.03	0.04	0.05	0.06	0.15	0.26	0.31	0.34	0.37
AID	0.01	0.02	0.03	0.04	0.06	0.02	0.05	0.08	0.10	0.12
MRD	0.01	0.03	0.06	0.08	0.11	0.10	0.22	0.27	0.31	0.33
SID	0.01	0.04	0.07	0.09	0.13	0.16	0.30	0.37	0.42	0.47

Table 4. The Effectiveness Evaluation on the MBPP Dataset

Methods	Incoder-1B					CodeGen-2B				
	pass@1	pass@3	pass@5	pass@7	pass@10	pass@1	pass@3	pass@5	pass@7	pass@10
Base	0.10	0.21	0.27	0.31	0.35	0.22	0.38	0.45	0.49	0.53
Insert_line	0.10	0.22	0.28	0.32	0.37	0.19	0.34	0.41	0.45	0.49
Comment	0.04	0.10	0.15	0.18	0.22	0.15	0.31	0.38	0.44	0.49
PPM-T	0.01	0.03	0.04	0.05	0.07	0.01	0.04	0.05	0.07	0.09
PPM-V	0.01	0.04	0.06	0.07	0.08	0.03	0.07	0.09	0.11	0.14
AID	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.02	0.03
MRD	0.01	0.03	0.04	0.06	0.08	0.01	0.03	0.05	0.07	0.10
SID	0.01	0.03	0.05	0.07	0.09	0.01	0.04	0.06	0.07	0.09
Methods	CodeGen2-1B					Santacoder				
	pass@1	pass@3	pass@5	pass@7	pass@10	pass@1	pass@3	pass@5	pass@7	pass@10
Base	0.11	0.21	0.28	0.32	0.37	0.24	0.41	0.48	0.52	0.55
Insert_line	0.10	0.20	0.26	0.29	0.33	0.18	0.34	0.41	0.46	0.50
Comment	0.06	0.14	0.19	0.22	0.26	0.23	0.40	0.48	0.53	0.58
PPM-T	0.01	0.04	0.06	0.08	0.10	0.02	0.06	0.08	0.10	0.12
PPM-V	0.01	0.04	0.06	0.07	0.09	0.04	0.08	0.10	0.12	0.14
AID	0.00	0.00	0.00	0.00	0.00	0.01	0.04	0.06	0.08	0.11
MRD	0.00	0.01	0.02	0.03	0.04	0.03	0.07	0.11	0.13	0.17
SID	0.00	0.01	0.02	0.03	0.04	0.03	0.07	0.10	0.13	0.16
Methods	Llama3-1b					Chatgpt				
	pass@1	pass@3	pass@5	pass@7	pass@10	pass@1	pass@3	pass@5	pass@7	pass@10
Base	0.24	0.43	0.50	0.55	0.59	0.76	0.83	0.85	0.86	0.87
Insert_line	0.23	0.39	0.46	0.49	0.53	0.76	0.82	0.84	0.85	0.86
Comment	0.18	0.34	0.41	0.46	0.50	0.76	0.82	0.84	0.85	0.86
PPM-T	0.03	0.08	0.12	0.15	0.19	0.41	0.50	0.52	0.54	0.55
PPM-V	0.03	0.08	0.11	0.13	0.16	0.32	0.40	0.42	0.44	0.46
AID	0.00	0.01	0.02	0.03	0.05	0.15	0.32	0.40	0.45	0.49
MRD	0.00	0.00	0.00	0.00	0.00	0.24	0.43	0.52	0.58	0.64
SID	0.00	0.01	0.01	0.01	0.02	0.30	0.52	0.62	0.68	0.74