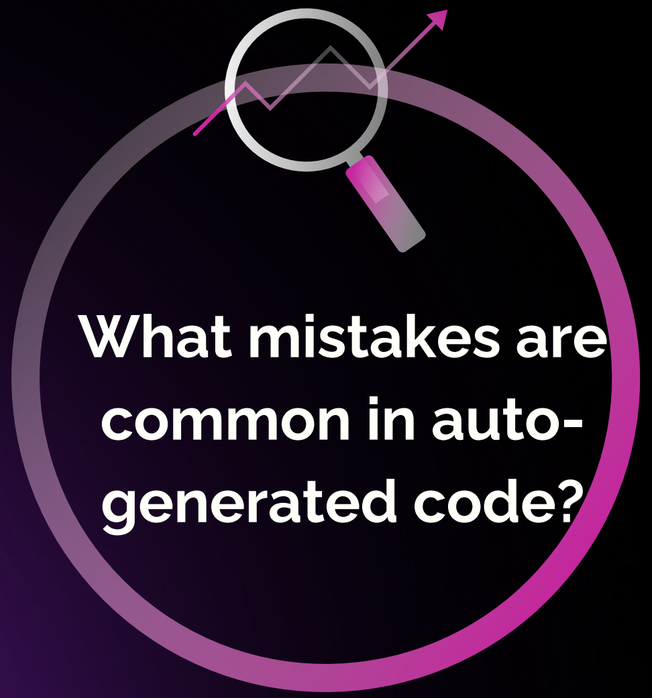
Automated repair of programs from large language models.

Fan, Zhiyu and Gao, Xiang and Mirchev, Martin and Roychoudhury, Abhik and Tan, Shin Hwei

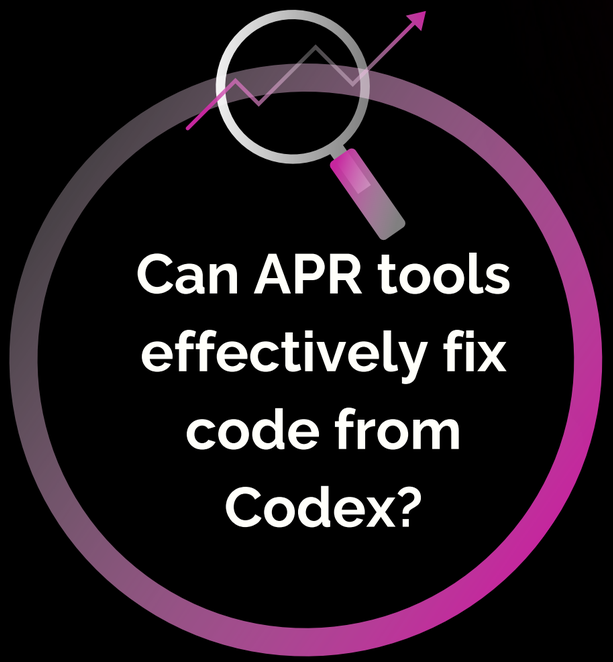
Motivations

- Low success of high quality code generation
- Lack awareness of semantics
- Programs don’t compile!
- No investigation of LLM-generated code by past works

Research Questions



What mistakes are common in auto-generated code?

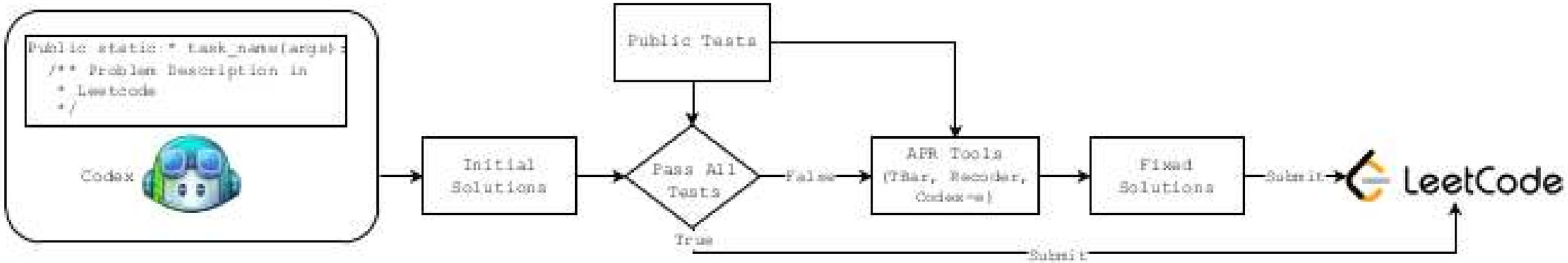


Can APR tools effectively fix code from Codex?

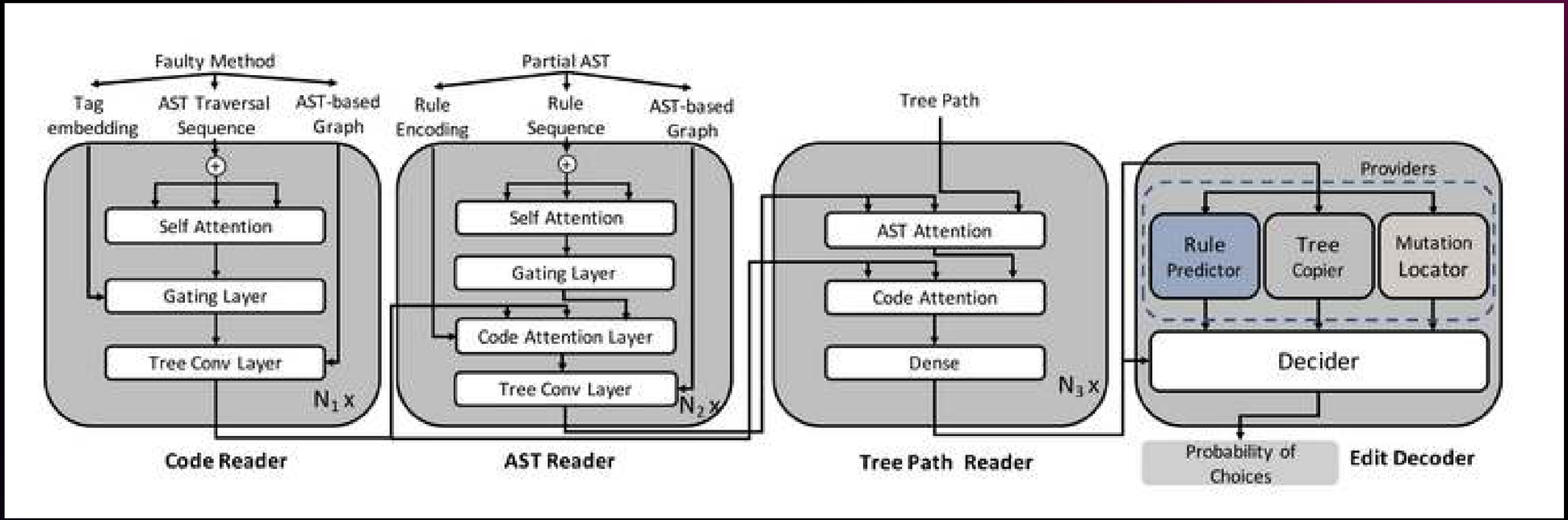
Can Codex edit mode fix program bugs?

Methodology

Codex + APR

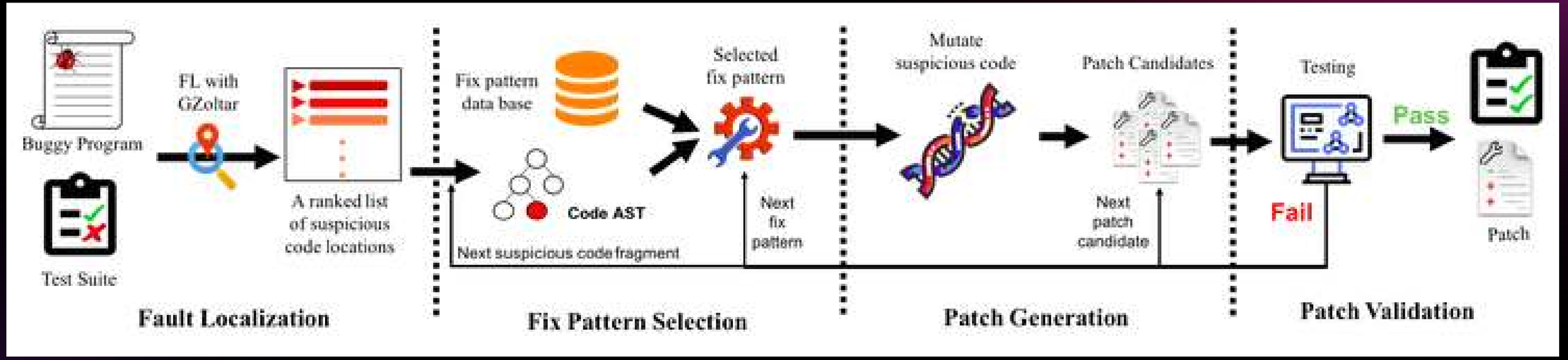


Recoder



https://github.com/pkuzqh/Recoder

TBar



https://github.com/TruX-DTF/TBar

Findings

Syntax Errors

incomplete code, or invoking undefined variables/functions/classes.

Misaligned Algorithm

Wrong algorithms
TLE
Naming conventions

Negative Symptoms

Anti-patterns

Similar Code Blocks

Irrelevant Helper Functions

Findings

```
1   public static int minimumOperations(int[] nums) {
2   /** You are given a 0-indexed array nums consisting
        of n positive integers. ...
3    * Return the minimum number of operations required
        to make the array alternating.*/
4     int n = nums.length;
5     int[] dp = new int[n];
6     dp[0] = 0; dp[1] = 1;
7     for (int i = 2; i < n; i++){
8       dp[i] = dp[i - 1] + 1;
9       if(nums[i] == nums[i - 2])
10        dp[i] = Math.min(dp[i - 2] + 1, dp[i]);
11    }
12    return dp[n - 1];
13  }
14  // Public Test 1:
15  // Input: nums = [3,1,3,2,4,3], Output: 3
16  // Public Test 2:
17  // Input: nums = [1,2,2,2,2], Output: 2
```

Fig. 2. An example of misaligned algorithm, taken from LeetCode programming task *minimumOperations* on weekly-contest-280.

Findings

TABLE III
THE NUMBER OF PATCHES AND FIXED TASKS PRODUCED BY TBAR AND RECODER (INCLUDE BOTH SINGLE-HUNK AND MULTI-HUNK)

Tool	Correct/Plausible patches		Correctly Fixed Tasks	
	easy	medium	easy	medium
TBar	6/16	3/22	3	3
Recoder	6/16	5/20	3	5

```
1   // task delete-characters-to-make-fancy-string
2   public static String makeFancyString(String s){...
3       if(...) {
4         sb.deleteCharAt(i);
5   -       i -= 2;
6   +       i -= 1; // constant mutation (S-C-3)
7       }...}
8   // task watering-plants
9   public static int wateringPlants(int[] plants, int
        capacity) {...
10    if (plants[i] > currWater) {
11      steps += (i - 1) * 2;
12  +   steps++; //add a statement (S-O-10)
13      ...} ...}
```

Fig. 4. Two incorrect solutions fixed by Recoder but not TBar.

Findings

TABLE IV

THE NUMBER OF CORRECTLY FIXED SOLUTIONS BY TBAR AND RECODER, REFER TABLE II FOR ABBREVIATION OF DEFECT CLASSIFICATION

Defect Sub-category	Total		TBar		Recoder	
	easy	medium	easy	medium	easy	medium
S-O	7	5	2	2	2	3
S-C	3	-	-	-	1	-
S-V	2	-	1	-	1	-
S-A	1	-	-	-	-	-
S-F	2	1	-	-	-	-
S-AS	8	1	-	-	-	1
S-DS	2	3	2	1	2	1
S-HO	5	5	1	-	-	-
Total (Single-Hunk)	30	15	6	3	6	5
M-S/M-U/M-L	31	31	-	-	-	-

Findings

TABLE VI

THE NUMBER OF INCORRECT SOLUTIONS THAT TBAR AND CODEX-E CAN PRODUCE ALL REQUIRED PATCH INGREDIENTS, REFER TABLE II FOR ABBREVIATION OF DEFECT CLASSIFICATION

Defect Sub-category	Total	TBar	Codex-e	TBar+Codex-e	TBar+Codex
S-HO	10	1	3	4	5
M-S	9	3	2	4	4
M-U	39	-	-	1	3
Total	58	4	5	9	12

Findings

TABLE V

THE NUMBER OF CORRECTLY FIXED SOLUTIONS USING CODEX-E, REFER TABLE II FOR ABBREVIATION OF DEFECT CLASSIFICATION

Defect Category	Sub -Category	Total		Codex-e^{bug}		Codex-e^{line}		Codex-e^{stm}	
		easy	medium	easy	medium	easy	medium	easy	medium
Single -Hunk	S-O	7	5	4	3	1	2	2	4
	S-C	3	-	-	-	1	-	1	-
	S-V	2	-	-	-	1	-	1	-
	S-A	1	-	1	-	-	-	-	-
	S-F	2	1	1	-	2	-	2	-
	S-AS	8	1	-	-	-	-	1	-
	S-DS	2	3	2	-	1	-	2	-
	S-HO	5	5	-	-	1	-	1	-
Total	-	30	15	8	3	7	2	10	4
Multi -Hunk	M-S	7	2	1	-	1	-	2	-
	M-U	19	20	1	2	-	1	-	-
	M-L	5	9	-	-	-	-	-	-
Total	-	31	31	2	2	1	1	2	-

Findings

```
1   public static int minSteps(String s, String t) {
2     int[] freq = new int[26];
3     for(char c : s.toCharArray())
4       freq[c - 'a']++;
5     int steps = 0;
6     for(char c : t.toCharArray())
7   -   if(freq[c - 'a'] == 0) // Fixed by TBar
8   -     steps++;
9   -   else
10        freq[c - 'a']--;
11  // Find in another candidate solution of minSteps
12  + for(int fr : freq)
13  +   steps += Math.abs(fr);
14    return steps;
15  }
```

Fig. 11. Obtaining patch ingredients from multiple candidate solutions

Strengths

Codex + APR

Codex's edit mode rivals TBar and Recoder

Relevant (LLM)

LLMDefects Dataset

Practical Insigths

APR integration, patch prioritization, and TDD

Weaknesses and Limitations

Test Suite Dependency

Fault Localization

Only Codex evaluated

Java LeetCode problems

Simple Bug Fixes

Small datasets

Conclusion

- Codex-generated code shares common defects with human-written code
- Codex-e outperforms traditional APR tools when guided by fault localization
- Fault Localization + LLM
- Complex bugs