金贤秀

软件工程师，不断学习

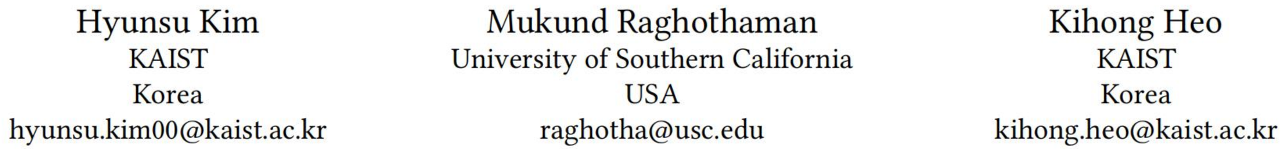
hyunsu.kim@furiosa.ai | Furiosa AI

论文汇报

现在

我是韩国首尔Furiosa AI的一名软件工程师。

《Learning Probabilistic Models for Static Analysis Alarms》	2022	ICSE (Best Artifact Awards)

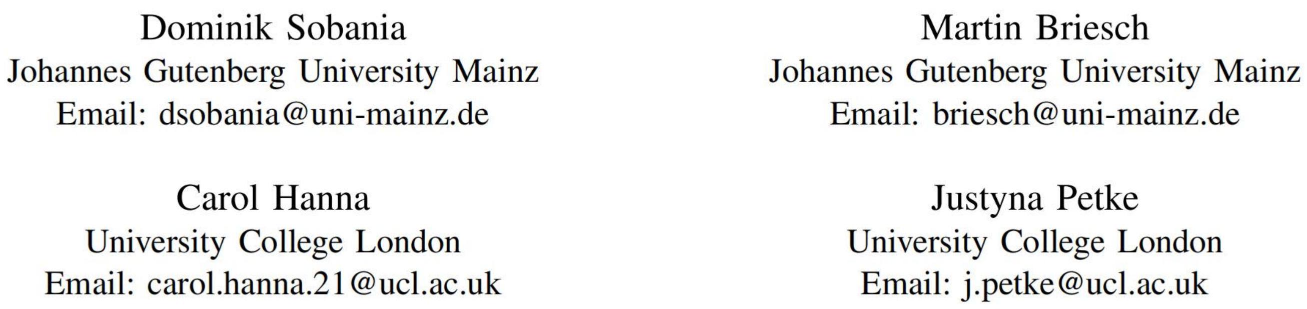


Hyunsu Kim
KAIST
Korea
hyunsu.kim00@kaist.ac.kr

Mukund Raghothaman
University of Southern California
USA
raghotha@usc.edu

Kihong Heo
KAIST
Korea
kihong.heo@kaist.ac.kr

《An Analysis of the Automatic Bug Fixing Performance of ChatGPT》	2023	CoRR



Dominik Sobania
Johannes Gutenberg University Mainz
Email: dsobania@uni-mainz.de

Martin Briesch
Johannes Gutenberg University Mainz
Email: briesch@uni-mainz.de

Carol Hanna
University College London
Email: carol.hanna.21@ucl.ac.uk

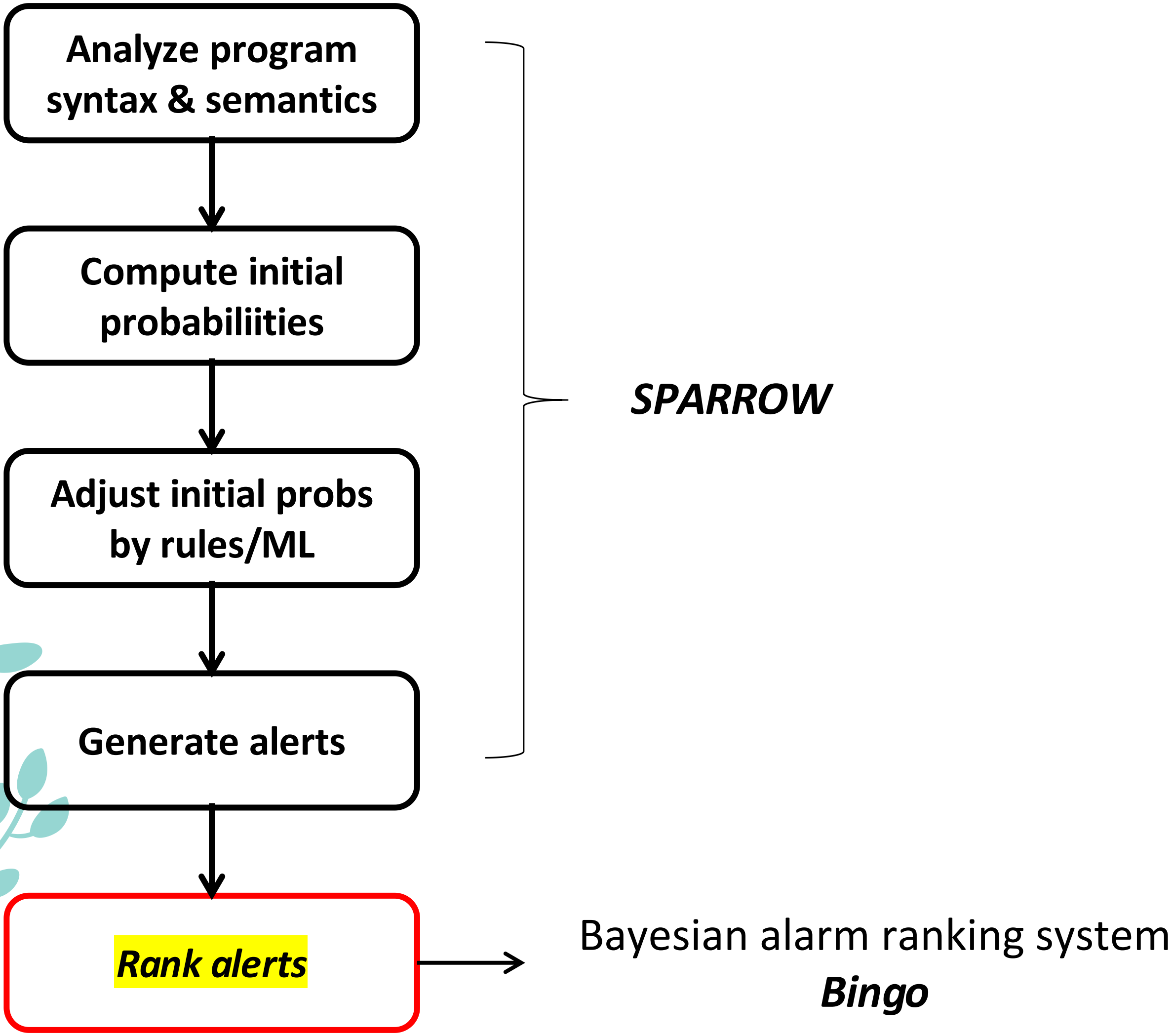
Justyna Petke
University College London
Email: j.petke@ucl.ac.uk

2023.3.9

Learning Probabilistic Models for Static Analysis Alarms

2022 ICSE

Static Analysis Alarms



Analyze program syntax & semantics

Compute initial probabiliities

Adjust initial probs by rules/ML

Generate alerts

Rank alerts

SPARROW

Bayesian alarm ranking system

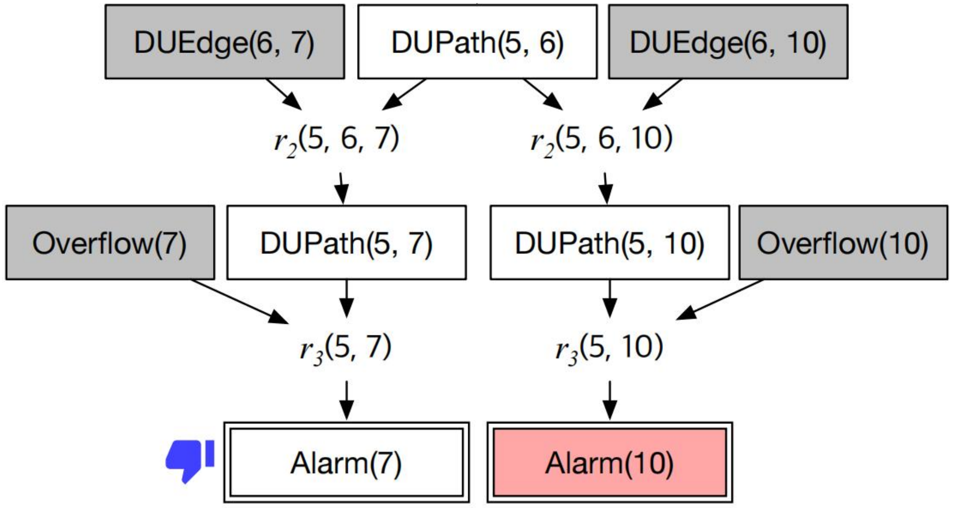
Bingo

False Generalization Problem

Rank	Alarm	Prob.		Rank	Alarm	Prob.
1	Alarm(7)	0.93			. . .	
2	Alarm(11)	0.92	→		. . .	
	. . .			461	Alarm(11)	0.41
9	Alarm(10)	0.91		462	Alarm(10)	0.40
	. . .				. . .	

Rank	Alarm	Prob.		Rank	Alarm	Prob.
1	Alarm(11)	0.97	→		. . .	
2	Alarm(10)	0.96		505	Alarm(10)	0.003
	. . .				. . .	

(a) Ranking changes after 166 and 187 interactions

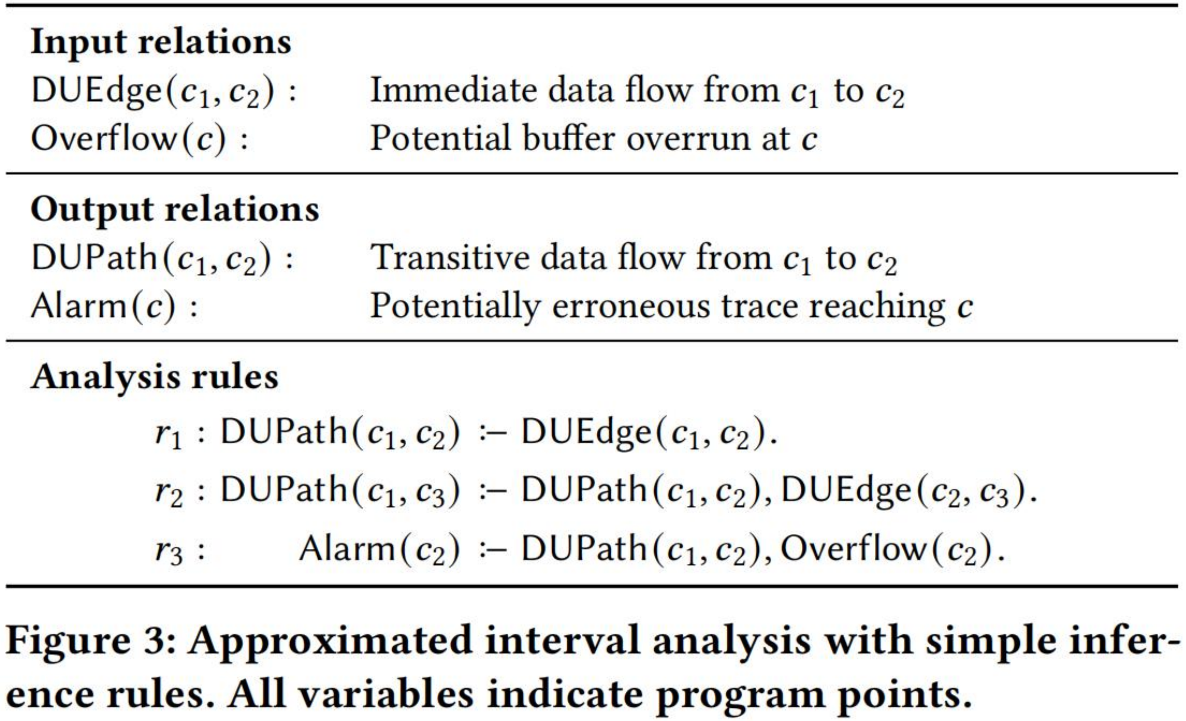


(b) Derivation graph

Learning Probabilistic Models for Static Analysis Alarms

2022 ICSE

False Generalization Problem



Input relations	
$\mathsf{DUEdge}(c_1, c_2)$:	Immediate data flow from c_1 to c_2
$\mathsf{Overflow}(c)$:	Potential buffer overrun at c
Output relations	
$\mathsf{DUPath}(c_1, c_2)$:	Transitive data flow from c_1 to c_2
$\mathsf{Alarm}(c)$:	Potentially erroneous trace reaching c

Analysis rules

$$r_1 : \mathsf{DUPath}(c_1, c_2) :- \mathsf{DUEdge}(c_1, c_2).$$

$$r_2 : \mathsf{DUPath}(c_1, c_3) :- \mathsf{DUPath}(c_1, c_2), \mathsf{DUEdge}(c_2, c_3).$$

$$r_3 : \mathsf{Alarm}(c_2) :- \mathsf{DUPath}(c_1, c_2), \mathsf{Overflow}(c_2).$$

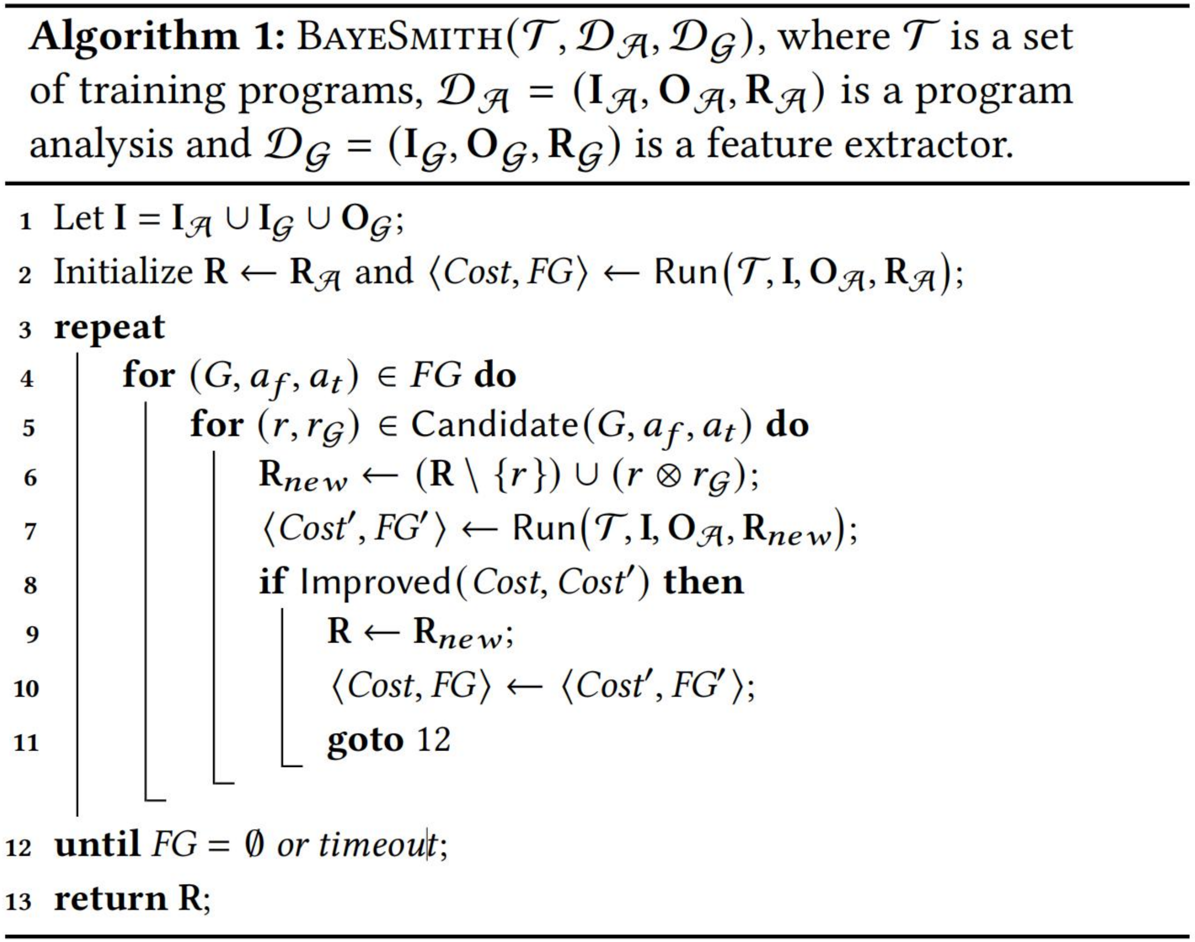
Figure 3: Approximated interval analysis with simple inference rules. All variables indicate program points.

Rule Refinement

$$r_{21} : \mathsf{DUPath}(c_1, c_3) :- \mathsf{DUPath}(c_1, c_2), \mathsf{Loop}(c_2), \mathsf{DUEdge}(c_2, c_3) \quad (5)$$

$$r_{22} : \mathsf{DUPath}(c_1, c_3) :- \mathsf{DUPath}(c_1, c_2), !\mathsf{Loop}(c_2), \mathsf{DUEdge}(c_2, c_3) \quad (6)$$

Learning Algorithm



Algorithm 1: BAYESMITH$(\mathcal{T}, \mathcal{D}_{\mathcal{A}}, \mathcal{D}_{\mathcal{G}})$, where $\mathcal{T}$ is a set of training programs, $\mathcal{D}_{\mathcal{A}} = (\mathbf{I}_{\mathcal{A}}, \mathbf{O}_{\mathcal{A}}, \mathbf{R}_{\mathcal{A}})$ is a program analysis and $\mathcal{D}_{\mathcal{G}} = (\mathbf{I}_{\mathcal{G}}, \mathbf{O}_{\mathcal{G}}, \mathbf{R}_{\mathcal{G}})$ is a feature extractor.

```
1  Let $\mathbf{I} = \mathbf{I}_{\mathcal{A}} \cup \mathbf{I}_{\mathcal{G}} \cup \mathbf{O}_{\mathcal{G}}$;
2  Initialize $\mathbf{R} \leftarrow \mathbf{R}_{\mathcal{A}}$ and $\langle Cost, FG \rangle \leftarrow \mathsf{Run}(\mathcal{T}, \mathbf{I}, \mathbf{O}_{\mathcal{A}}, \mathbf{R}_{\mathcal{A}})$;
3  repeat
4      for $(G, a_f, a_t) \in FG$ do
5          for $(r, r_G) \in \mathsf{Candidate}(G, a_f, a_t)$ do
6              $\mathbf{R}_{new} \leftarrow (\mathbf{R} \setminus \{r\}) \cup (r \otimes r_G)$;
7              $\langle Cost', FG' \rangle \leftarrow \mathsf{Run}(\mathcal{T}, \mathbf{I}, \mathbf{O}_{\mathcal{A}}, \mathbf{R}_{new})$;
8              if $\mathsf{Improved}(Cost, Cost')$ then
9                  $\mathbf{R} \leftarrow \mathbf{R}_{new}$;
10                 $\langle Cost, FG \rangle \leftarrow \langle Cost', FG' \rangle$;
11                 goto 12
12 until $FG = \emptyset$ or timeout;
13 return $\mathbf{R}$;
```

Learning Probabilistic Models for Static Analysis Alarms

2022 ICSE

Overflow

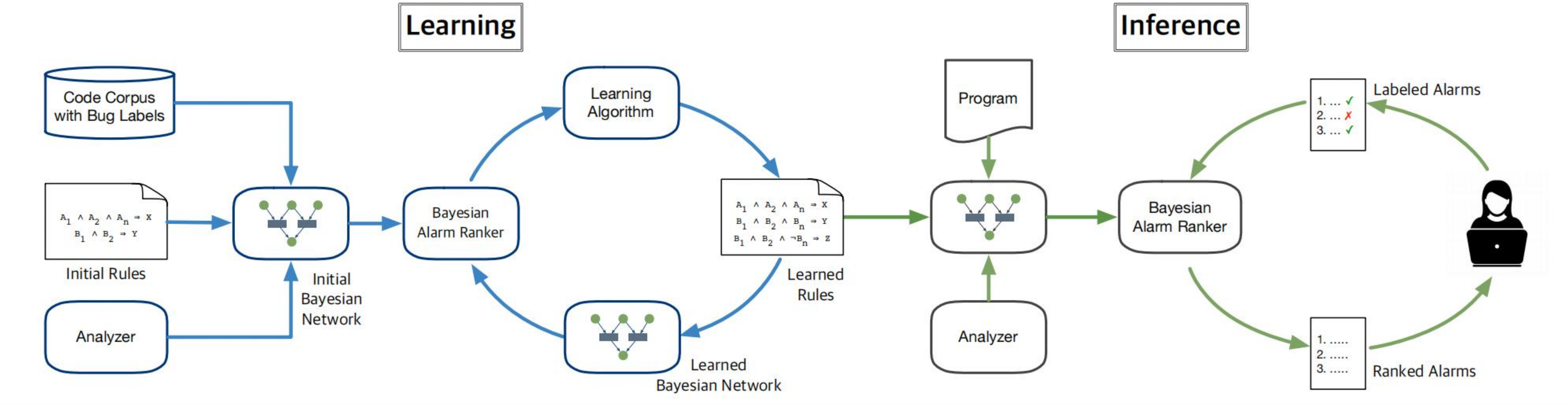


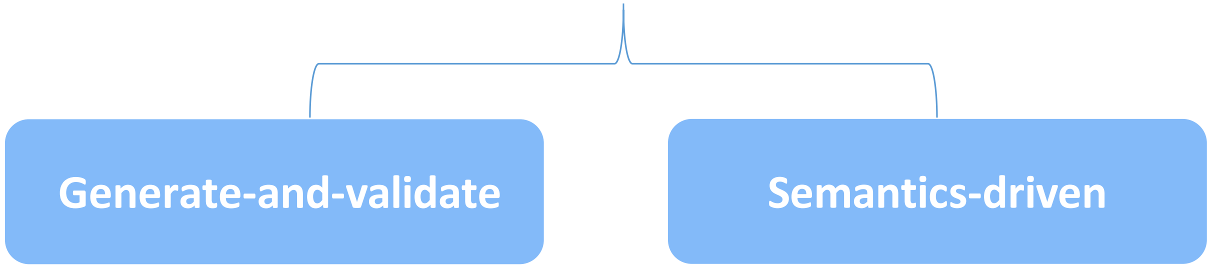
Figure 4: System overview

…

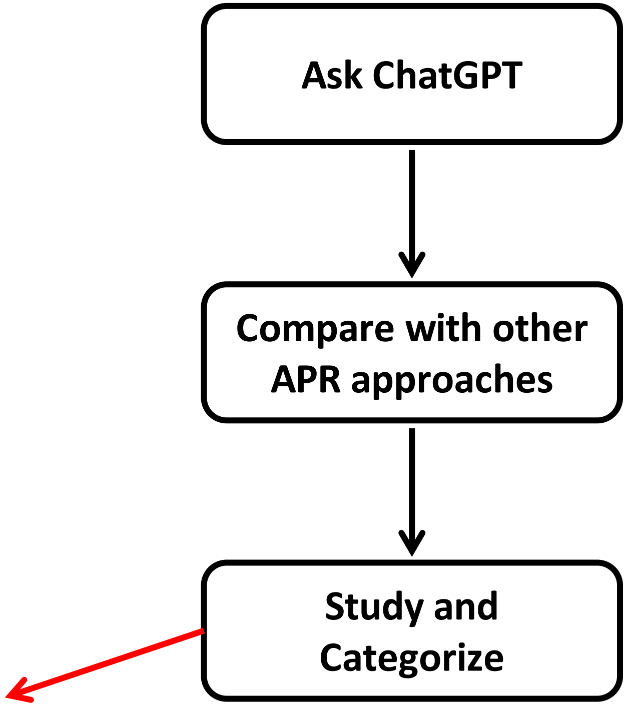
An Analysis of the Automatic Bug Fixing Performance of ChatGPT

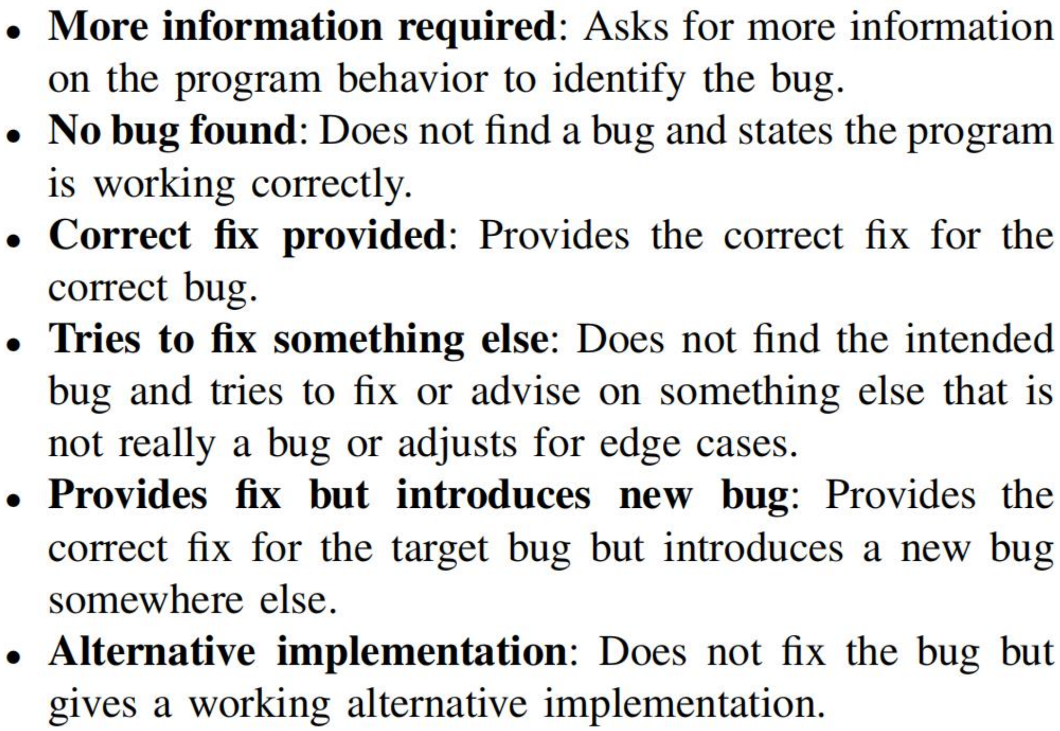
2023 CoRR

APR

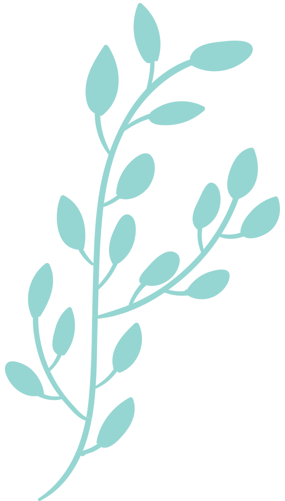


Overflow





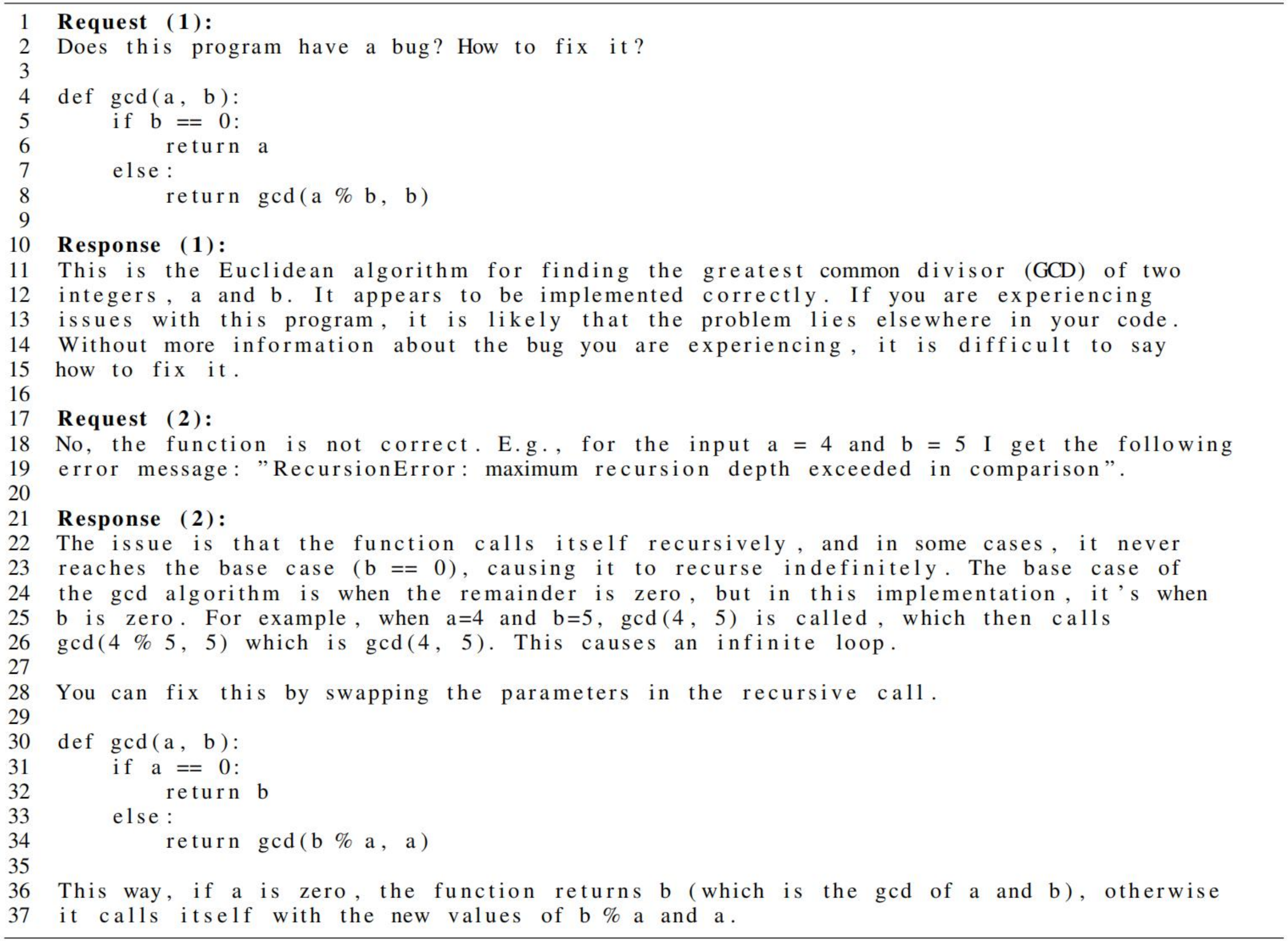
- **More information required**: Asks for more information on the program behavior to identify the bug.
- **No bug found**: Does not find a bug and states the program is working correctly.
- **Correct fix provided**: Provides the correct fix for the correct bug.
- **Tries to fix something else**: Does not find the intended bug and tries to fix or advise on something else that is not really a bug or adjusts for edge cases.
- **Provides fix but introduces new bug**: Provides the correct fix for the target bug but introduces a new bug somewhere else.
- **Alternative implementation**: Does not fix the bug but gives a working alternative implementation.



An Analysis of the Automatic Bug Fixing Performance of ChatGPT

2023 CoRR

Dialogue S



Request (1):
Does this program have a bug? How to fix it?

```
def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(a % b, b)
```

Response (1):
This is the Euclidean algorithm for finding the greatest common divisor (GCD) of two integers, a and b. It appears to be implemented correctly. If you are experiencing issues with this program, it is likely that the problem lies elsewhere in your code. Without more information about the bug you are experiencing, it is difficult to say how to fix it.

Request (2):
No, the function is not correct. E.g., for the input a = 4 and b = 5 I get the following error message: "RecursionError: maximum recursion depth exceeded in comparison".

Response (2):
The issue is that the function calls itself recursively, and in some cases, it never reaches the base case (b == 0), causing it to recurse indefinitely. The base case of the gcd algorithm is when the remainder is zero, but in this implementation, it's when b is zero. For example, when a=4 and b=5, gcd(4, 5) is called, which then calls gcd(4 % 5, 5) which is gcd(4, 5). This causes an infinite loop.

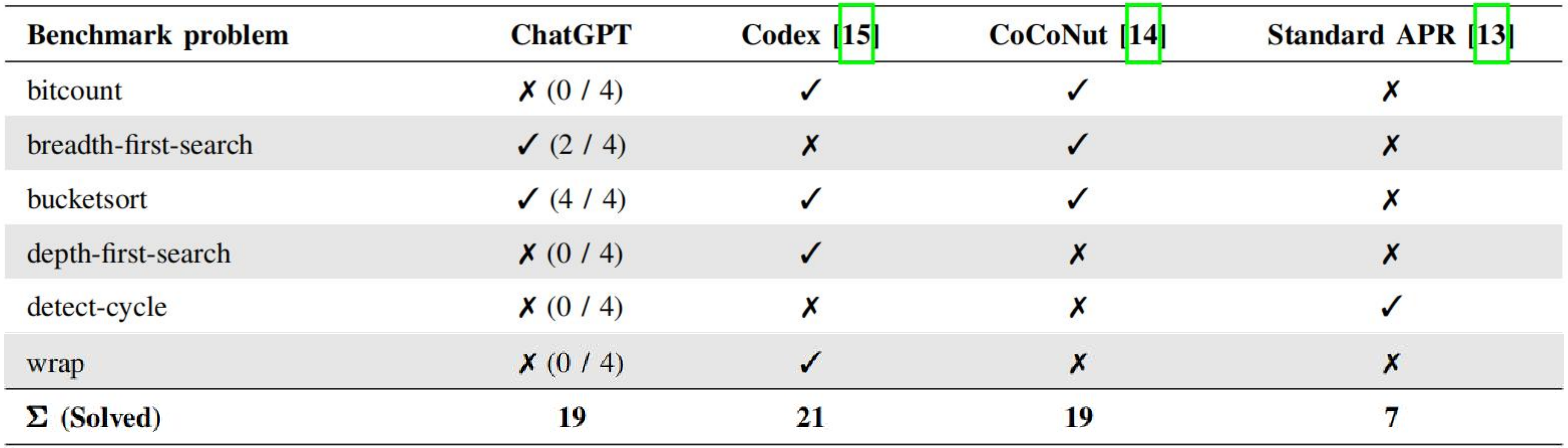
You can fix this by swapping the parameters in the recursive call.

```
def gcd(a, b):
    if a == 0:
        return b
    else:
        return gcd(b % a, a)
```

This way, if a is zero, the function returns b (which is the gcd of a and b), otherwise it calls itself with the new values of b % a and a.

An Analysis of the Automatic Bug Fixing Performance of ChatGPT

2023 CoRR



Benchmark problem	ChatGPT	Codex [15]	CoCoNut [14]	Standard APR [13]
bitcount	✗ (0 / 4)	✓	✓	✗
breadth-first-search	✓ (2 / 4)	✗	✓	✗
bucketsort	✓ (4 / 4)	✓	✓	✗
depth-first-search	✗ (0 / 4)	✓	✗	✗
detect-cycle	✗ (0 / 4)	✗	✗	✓
wrap	✗ (0 / 4)	✓	✗	✗
Σ (Solved)	**19**	**21**	**19**	**7**

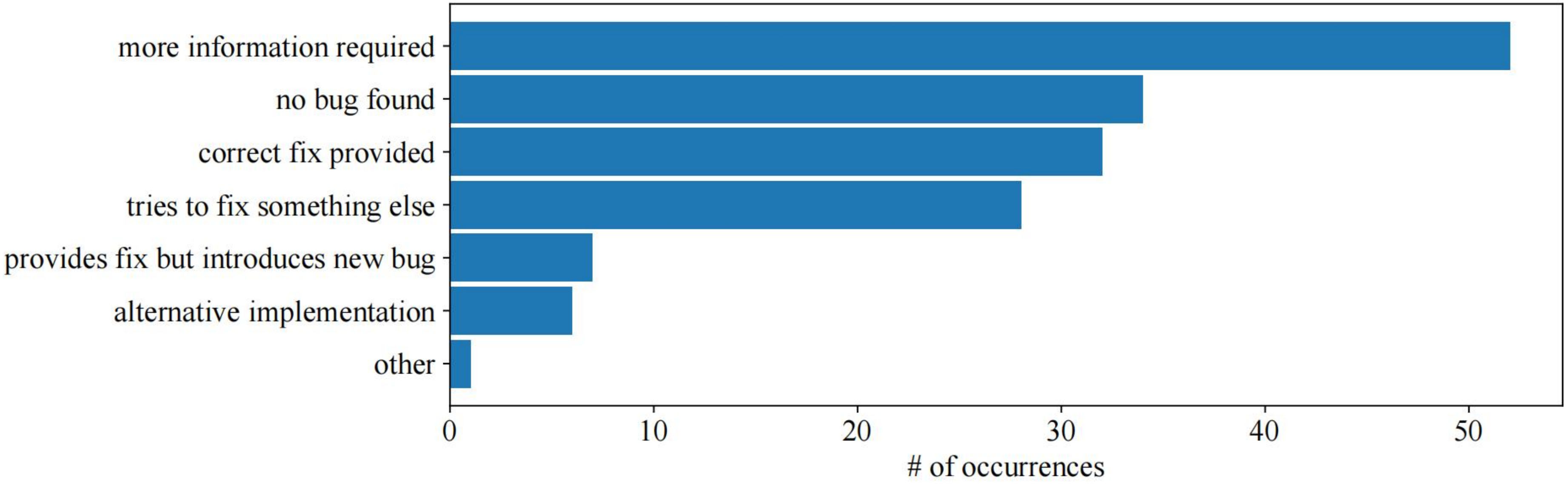
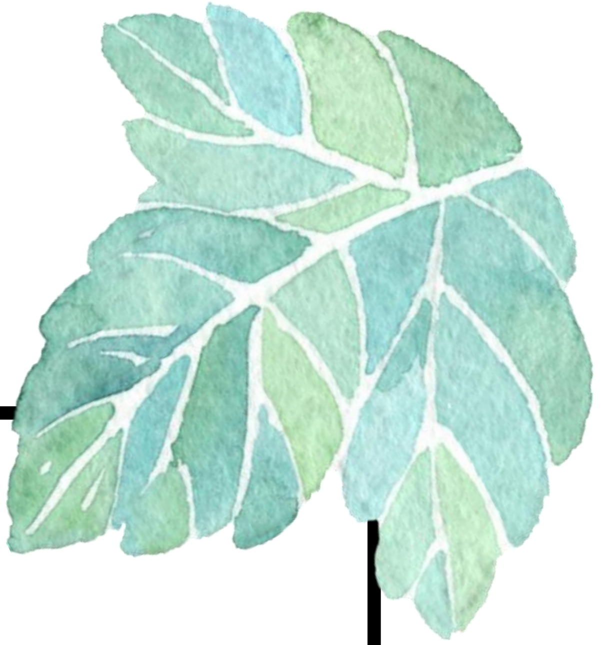


Fig. 2: Number of occurrences of identified classes of ChatGPT answers given for the problems from QuixBugs.

谢谢聆听



演讲完毕