MENTOR: Fixing Introductory Programming Assignments With Formula-Based Fault Localization and LLM-Driven Program Repair

Pedro Orvalho [1]

[1]Instituto de Investigación en Inteligencia Artificial (IIIA), Consejo Superior de Investigaciones Científicas (CSIC), Barcelona, Spain

Work done as a doctoral researcher at:
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
CIIRC, Czech Technical University in Prague, Czechia

INESC-ID Best PhD Student 2025 Award Ceremony
Lisbon, 19 December 2025

IIIA-CSIC INSTITUT D'INVESTIGACIÓ EN INTEL·LIGÈNCIA ARTIFICIAL
CSIC CONSEJO SUPERIOR DE INVESTIGACIONES CIENTÍFICAS
TÉCNICO LISBOA
inesc id lisboa
CZECH INSTITUTE OF INFORMATICS ROBOTICS AND CYBERNETICS CTU IN PRAGUE
fct Fundação para a Ciência e a Tecnologia

Motivation

- The increasing demand for programming education has given rise to all kinds of online evaluations focused on introductory programming assignments (IPAs):

Motivation

- The increasing demand for programming education has given rise to all kinds of online evaluations focused on introductory programming assignments (IPAs):
    - MIT's MOOC, Introduction to CS, **reached 1.2 M enrollments** in 2018;

Motivation

- The increasing demand for programming education has given rise to all kinds of online evaluations focused on introductory programming assignments (IPAs):
  - MIT's MOOC, Introduction to CS, **reached 1.2 M enrollments** in 2018;
  - In 2020, Stanford's CS MOOC had **more than 10K students**.

Motivation

- In these courses it is a challenge to **provide personalized feedback to students**.

Motivation

- In these courses it is a challenge to **provide personalized feedback to students**.
- Providing feedback in IPAs **requires substantial time and effort by faculty**.

Automated Program Repair (APR)

Given a buggy program P_o and a set of input-output examples T (test suite).

Automated Program Repair (APR)

Given a buggy program P_o and a set of input-output examples T (test suite).

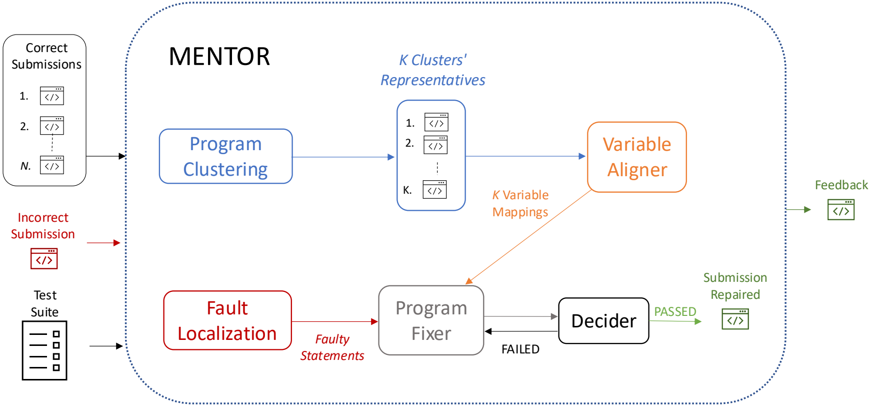
The goal of *Automated Program Repair* is to find a program P_f by **semantically change a subset S_1 of P_o's statements** ($S_1 \subseteq P_o$) for another set of statements S_2, s.t.,

$$P_f = ((P_o \setminus S_1) \cup S_2)$$

and

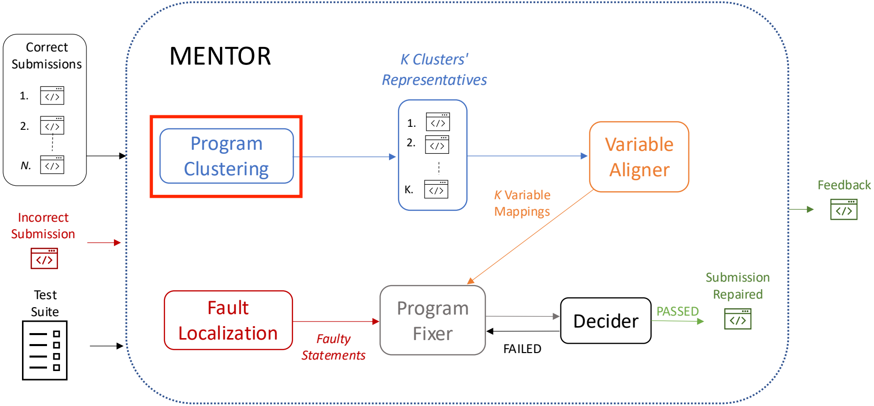
$$\forall (t_{in}^i, t_{out}^i) \in T \ : \ P_f(t_{in}^i) = t_{out}^i$$

MENTOR: Automated Feedback for Introductory Programming Exercises



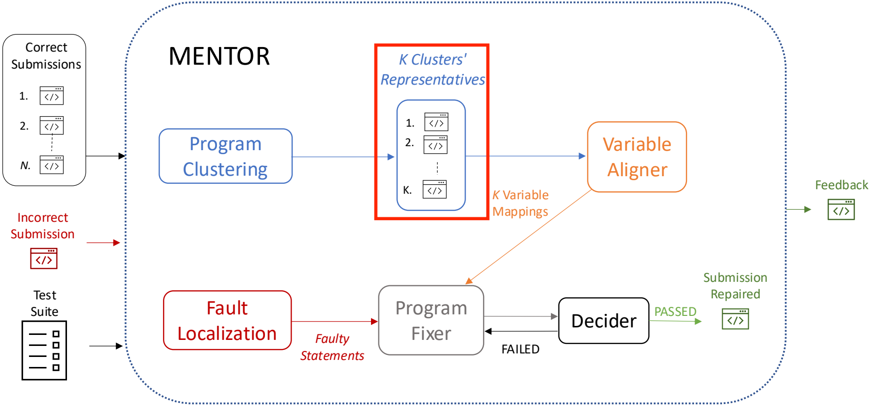
- P. Orvalho, M. Janota, and V. Manquinho. MENTOR: Fixing Introductory Programming Assignments with Formula-Based Fault Localization and LLM-Driven Program Repair. In **JSS 2026**.

MENTOR



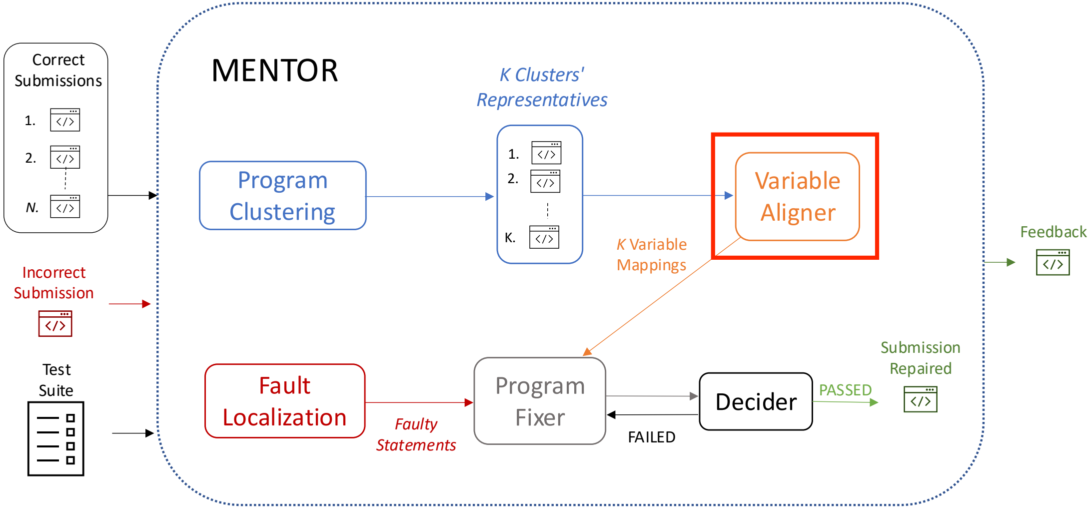
- P. Orvalho, M. Janota, and V. Manquinho. InvAASTCluster: On Applying Invariant-Based Program Clustering to Introductory Programming Assignments. In **JSS 2025**.

MENTOR



- P. Orvalho, M. Janota, and V. Manquinho. InvAASTCluster: On Applying Invariant-Based Program Clustering to Introductory Programming Assignments. In **JSS 2025**.

MENTOR


Correct Submissions
1.
2.
N.
Incorrect Submission
Test Suite
MENTOR
K Clusters' Representatives
Program Clustering
1.
2.
K.
Variable Aligner
K Variable Mappings
Fault Localization
Faulty Statements
Program Fixer
Decider
FAILED
PASSED
Submission Repaired
Feedback

Variable Mapping

- P. Orvalho, J. Piepenbrock, M. Janota, and V. Manquinho. Graph Neural Networks For Mapping Variables Between Programs. In **ECAI 2023**.
- P. Orvalho, M. Janota, and V. Manquinho. MultIPAs: Applying Program Transformations to Introductory Programming Assignments for Data Augmentation. In **ESEC/FSE 2022**.

Variable Mapping - Motivation

1: Function that finds and returns the maximum number among `n1`, `n2` and `n3`.

```
int max(int n1, int n2, int n3)
{
    int m = n1 > n2 ? n1 : n2;
    return n3 > m ? n3 : m;
}
```

2: Function that finds and returns the maximum number among `x`, `y` and `z`.

```
int max(int x, int y, int z){
    int m = 0;
    m = x > m ? x : m;
    m = y > m ? y : m;
    return z > m ? z : m;
}
```

Variable Mapping: $\{\mathtt{m} : \mathtt{m}; \mathtt{n1} : \mathtt{x}; \mathtt{n2} : \mathtt{y}; \mathtt{n3} : \mathtt{z}\}$.

Program Representation

3: An expression that uses int variables `a` and `b`, previously declared in the program.

```
1  {
2      // a and b are ints
3      a = a - b;
4  }
```

Types of edges:

AST

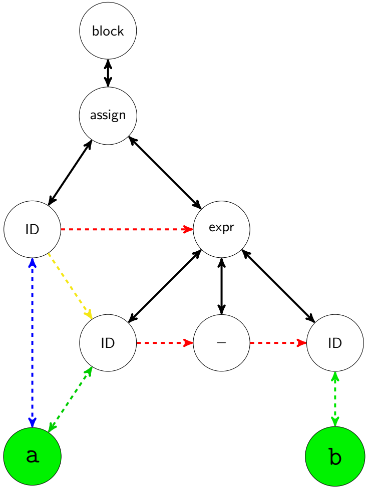
Read

Write

Sibling

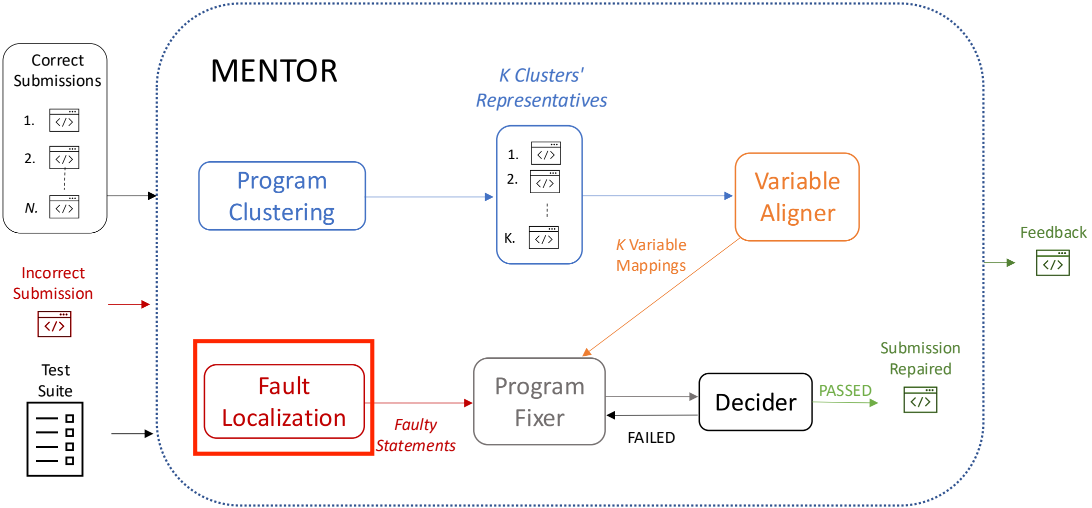
Chronological

Variable Node



(a) Our program representation.

MENTOR



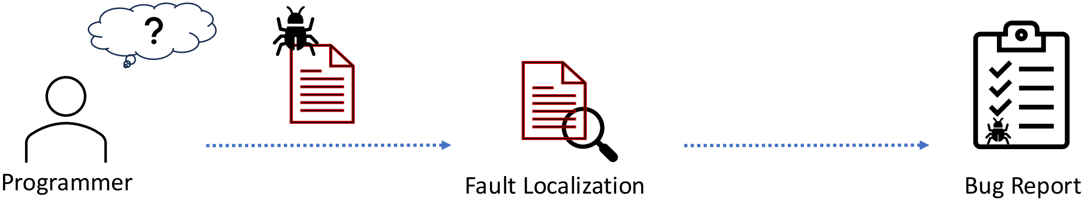
Correct
Submissions
1.
2.
N.
Incorrect
Submission
Test
Suite
MENTOR
K Clusters'
Representatives
1.
2.
K.
Program
Clustering
Variable
Aligner
K Variable
Mappings
Feedback
Fault
Localization
Faulty
Statements
Program
Fixer
Decider
PASSED
FAILED
Submission
Repaired

Fault Localization

- P. Orvalho, M. Janota, and V. Manquinho. CFaults: Model-Based Diagnosis for Fault Localization in C with Multiple Test Cases. In **FM 2024**.

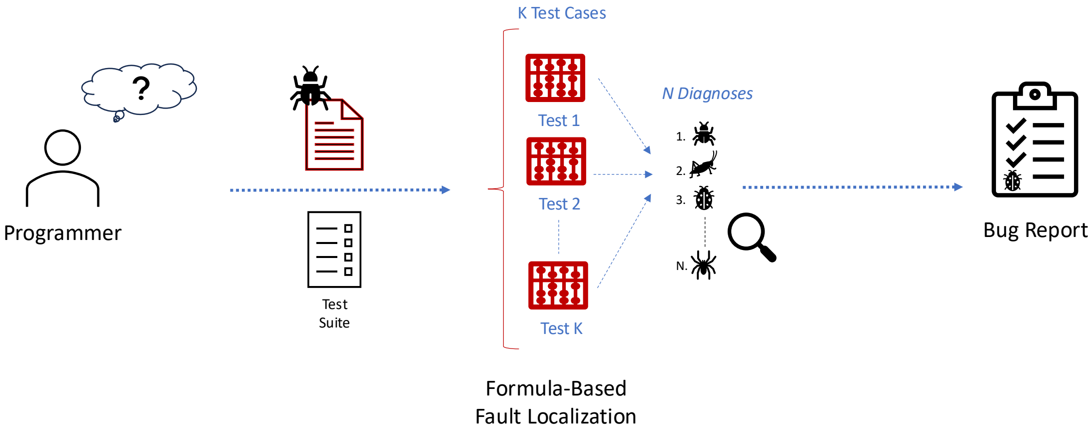
Fault Localization

- Given a buggy program, *fault localization (FL)* involves identifying locations in the program that could cause a faulty behaviour (bug).

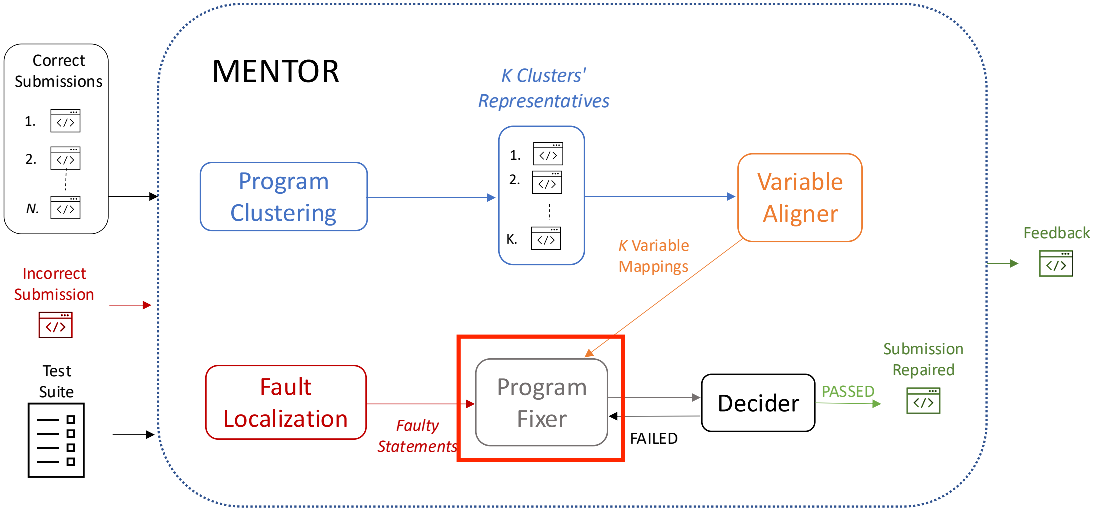


Formula-Based Fault Localization (FBFL)

- FBFL methods encode the localization problem into **several optimization problems** to identify a minimal set of bugs (diagnoses).



MENTOR


Correct
Submissions
1.
2.
N.
Incorrect
Submission
Test
Suite
MENTOR
Program
Clustering
K Clusters'
Representatives
1.
2.
K.
Variable
Aligner
K Variable
Mappings
Fault
Localization
Faulty
Statements
Program
Fixer
Decider
PASSED
FAILED
Submission
Repaired
Feedback

Program Repair

- P. Orvalho, M. Janota, and V. Manquinho. Counterexample Guided Program Repair Using Zero-Shot Learning and MaxSAT-based Fault Localization. In **AAAI 2025**.

Motivation

4: Semantically incorrect program. Faults: {4,8}.

```c
1  int main(){ //finds max of 3 nums
2      int f,s,t;
3      scanf("%d%d%d",&f,&s,&t);
4      if (f < s && f >= t)
5          printf("%d",f);
6      else if (s > f && s >= t)
7          printf("%d",s);
8      else if (t < f && t < s)
9          printf("%d",t);
10
11     return 0;
12 }
```

Motivation

5: Semantically incorrect program. Faults: {4,8}.

```
1  int main(){ //finds max of 3 nums
2      int f,s,t;
3      scanf("%d%d%d",&f,&s,&t);
4      if (f < s && f >= t)
5          printf("%d",f);
6      else if (s > f && s >= t)
7          printf("%d",s);
8      else if (t < f && t < s)
9          printf("%d",t);
10
11     return 0;
12 }
```

Motivation

6: Semantically incorrect program. Faults: {4,8}.

```
1  int main(){ //finds max of 3 nums
2      int f,s,t;
3      scanf("%d%d%d",&f,&s,&t);
4      if (f < s && f >= t)
5          printf("%d",f);
6      else if (s > f && s >= t)
7          printf("%d",s);
8      else if (t < f && t < s)
9          printf("%d",t);
10
11     return 0;
12 }
```

LLMs for code (LLMCs)

- GRANITE and CODEGEMMA **cannot fix** the buggy program within 90 secs;

Motivation

7: Semantically incorrect program. Faults: {4,8}.

```
1   int main(){ //finds max of 3 nums
2       int f,s,t;
3       scanf("%d%d%d",&f,&s,&t);
4       if (f < s && f >= t)
5           printf("%d",f);
6       else if (s > f && s >= t)
7           printf("%d",s);
8       else if (t < f && t < s)
9           printf("%d",t);
10
11      return 0;
12  }
```

LLMs for code (LLMCs)

- Granite and CodeGemma **cannot fix** the buggy program within 90 secs;
- Even if we provide the assignment's **description and IO tests**.

Program Sketches

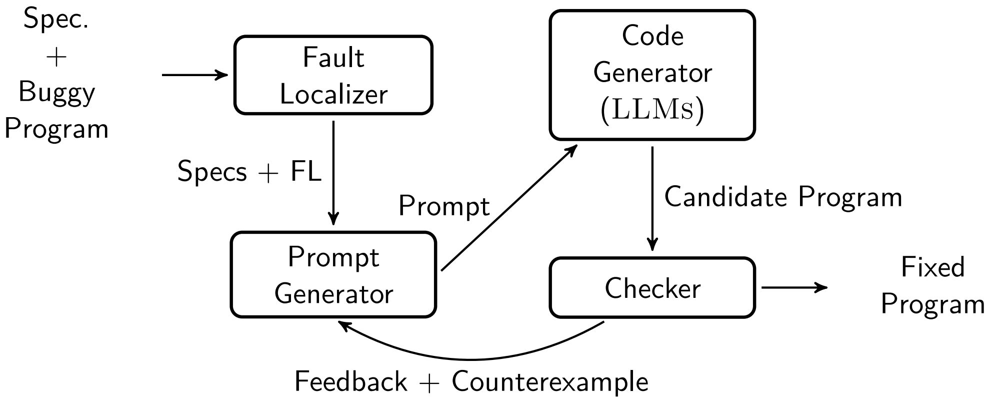
8: Semantically incorrect program. Faults :{4,8}.

```
1  int main(){ //finds max of 3 nums
2      int f,s,t;
3      scanf("%d%d%d",&f,&s,&t);
4      if (f < s && f >= t)
5          printf("%d",f);
6      else if (s > f && s >= t)
7          printf("%d",s);
8      else if (t < f && t < s)
9          printf("%d",t);
10
11     return 0;
12 }
```

9: Program sketch with holes.

```
1  int main(){
2      int f,s,t;
3      scanf("%d%d%d",&f,&s,&t);
4      @ HOLE 1 @
5          printf("%d",f);
6      else if (s > f && s >= t)
7          printf("%d",s);
8      @ HOLE 2 @
9          printf("%d",t);
10
11     return 0;
12 }
```

Counterexample Guided Automated Repair

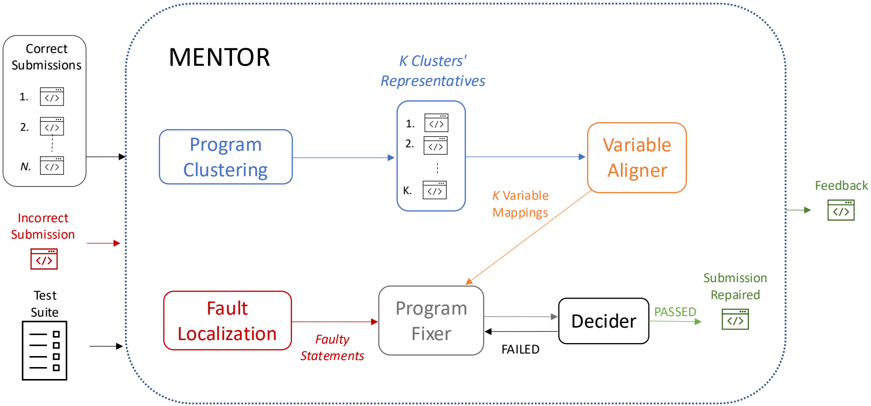


LLM-Driven APR with CFaults

	Prompt Configurations			
LLMs	**De-TS**	**De-TS-CE**	**Sk_De-TS**	**Sk_De-TS-CE**
CodeGemma	41.7%	42.3%	47.7%	**48.1%**
CodeLlama	34.4%	34.9%	**40.0%**	39.2%
Gemma	34.7%	34.4%	37.2%	**37.3%**
Granite	43.7%	43.6%	**48.3%**	47.6%
Llama3	39.4%	41.2%	40.4%	**41.3%**
Phi3	34.5%	34.2%	**38.2%**	37.4%
Verifix	6.3%			
Clara	34.6%			

Table 1: The number of programs fixed by each LLM under various configurations. Mapping abbreviations to configuration names: **De** - IPA *Description*, **TS** - *Test Suite*, **CE** - *Counterexample*, **SK** - *Sketches*.

MENTOR: Automated Feedback for Introductory Programming Exercises



- P. Orvalho, M. Janota, and V. Manquinho. MENTOR: Fixing Introductory Programming Assignments with Formula-Based Fault Localization and LLM-Driven Program Repair. In **JSS 2026**.

Takeaway Message

- We tackle the APR problem using an **LLM-Driven Counterexample Guided Inductive Synthesis (CEGIS) approach** [Solar-Lezama et al., 2006];

Takeaway Message

- We tackle the APR problem using an **LLM-Driven Counterexample Guided Inductive Synthesis (CEGIS) approach** [Solar-Lezama et al., 2006];
- We employ **Logic-based Fault Localization to guide and minimize LLMs' patches** to incorrect programs by feeding them bug-free program sketches;

Takeaway Message

- We tackle the APR problem using an **LLM-Driven Counterexample Guided Inductive Synthesis (CEGIS) approach** [Solar-Lezama et al., 2006];

- We employ **Logic-based Fault Localization to guide and minimize LLMs' patches** to incorrect programs by feeding them bug-free program sketches;

- With our approach **all six evaluated LLMs fix more programs and produce smaller patches** than other configurations and symbolic tools;

Takeaway Message

- We tackle the APR problem using an **LLM-Driven Counterexample Guided Inductive Synthesis (CEGIS) approach** [Solar-Lezama et al., 2006];
- We employ **Logic-based Fault Localization to guide and minimize LLMs' patches** to incorrect programs by feeding them bug-free program sketches;
- With our approach **all six evaluated LLMs fix more programs and produce smaller patches** than other configurations and symbolic tools;
- All our code is available on GitHub and on Zenodo.

Obrigado!
Thank you!

Obrigado!
Thank you!

My sincere thanks to INESC-ID,

Obrigado!
Thank you!

My sincere thanks to INESC-ID,
and to my advisors, Vasco and Mikoláš!

Pedro Orvalho

Thank you!

https://pmorvalho.github.io

References

Reiter, Raymond (1987)
A Theory of Diagnosis from First Principles.
Artif. Intell. 1987.

Do, Hyunsook and Elbaum, Sebastian G. and Rothermel, Gregg (2005)
Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and its Potential Impact.
Empir. Softw. Eng. 2005.

Jose, Manu and Majumdar, Rupak (2011)
Cause clue clauses: error localization using maximum satisfiability.
PLDI 2011.

Armando Solar-Lezama and Liviu Tancau and Rastislav Bodík and Sanjit A. Seshia and Vijay A. Saraswat (2018)
Combinatorial sketching for finite programs.
ASPLOS 2006.

References

Ignatiev, Alexey and Morgado, António and Weissenbacher, Georg and Marques-Silva, João (2019)
Model-Based Diagnosis with Multiple Observations.
IJCAI 2019.

Orvalho, Pedro and Janota, Mikolas and Manquinho, Vasco (2024)
C-Pack of IPAs: A C90 Program Benchmark of Introductory Programming Assignments.
Automated Program Repair (APR) 2024.

Lamraoui, Si-Mohamed and Nakajima, Shin (2016)
A Formula-based Approach for Automatic Fault Localization of Multi-fault Programs.
J. Inf. Process. 24(1), 88 – 98.

The Guardian UK - Crowdstrike Meltdown
https://www.theguardian.com/technology/article/2024/jul/24/crowdstrike-outage-companies-cost.
The Guardian UK.

Ahmed, Umair Z and Fan, Zhiyu and Yi, Jooyong and Al-Bataineh, Omar I and Roychoudhury, Abhik (2022)
Verifix: Verified repair of programming assignments.
TOSEM 22 12(3), 45 – 678.

References

Orvalho, Pedro and Janota, Mikoláš and Manquinho, Vasco (2022)
MultIPAs: Applying Program Transformations to Introductory Programming Assignments for Data Augmentation.
ESEC/FSE 2022.

Gulwani, Sumit and Radiček, Ivan and Zuleger, Florian (2018)
Automated clustering and program repair for introductory programming assignments.
PLDI 18 52(4), 465 – 480.

Orvalho, Pedro and Janota, Mikolas and Manquinho, Vasco (2024)
CFaults: Model-Based Diagnosis for Fault Localization in C with Multiple Test Cases.
Formal Methods (FM) 2024.

Orvalho, Pedro and Janota, Mikolas and Manquinho, Vasco (2025)
Counterexample Guided Program Repair Using Zero-Shot Learning and MaxSAT-based Fault Localization.
AAAI 2025.