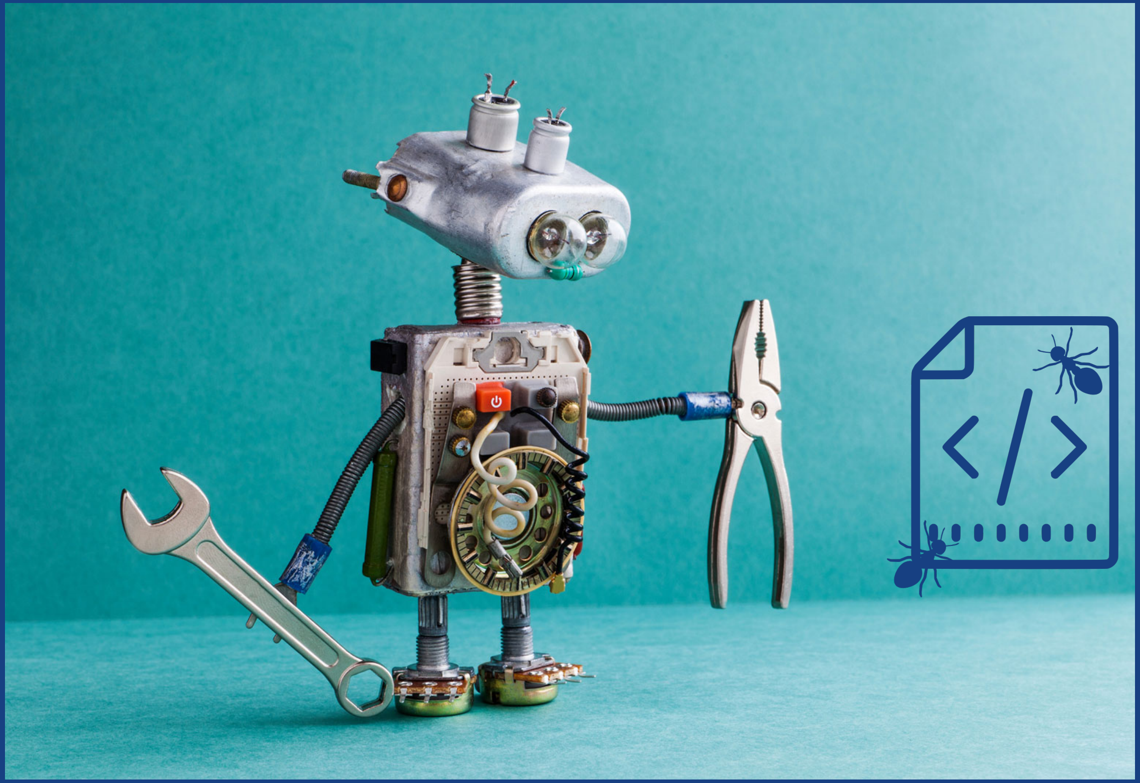
Is this Change the Answer to that Problem? Correlating Descriptions of Bug and Code Changes for Evaluating Patch Correctness

Haoye Tian[1], Xunzhu Tang[1], Andrew Habib[1], Shangwen Wang[2], Kui Liu[3], Xin Xia[3], Jacques Klein[1], Tegawendé F Bissyandé[1]

[1] Univeristy of Luxembourg, [2] National University of Defense - China, [3] Huawei

ASE'22 – Oct. 10 -14, 2022

Automated Program Repair (APR)



Automated Program Repair (APR)

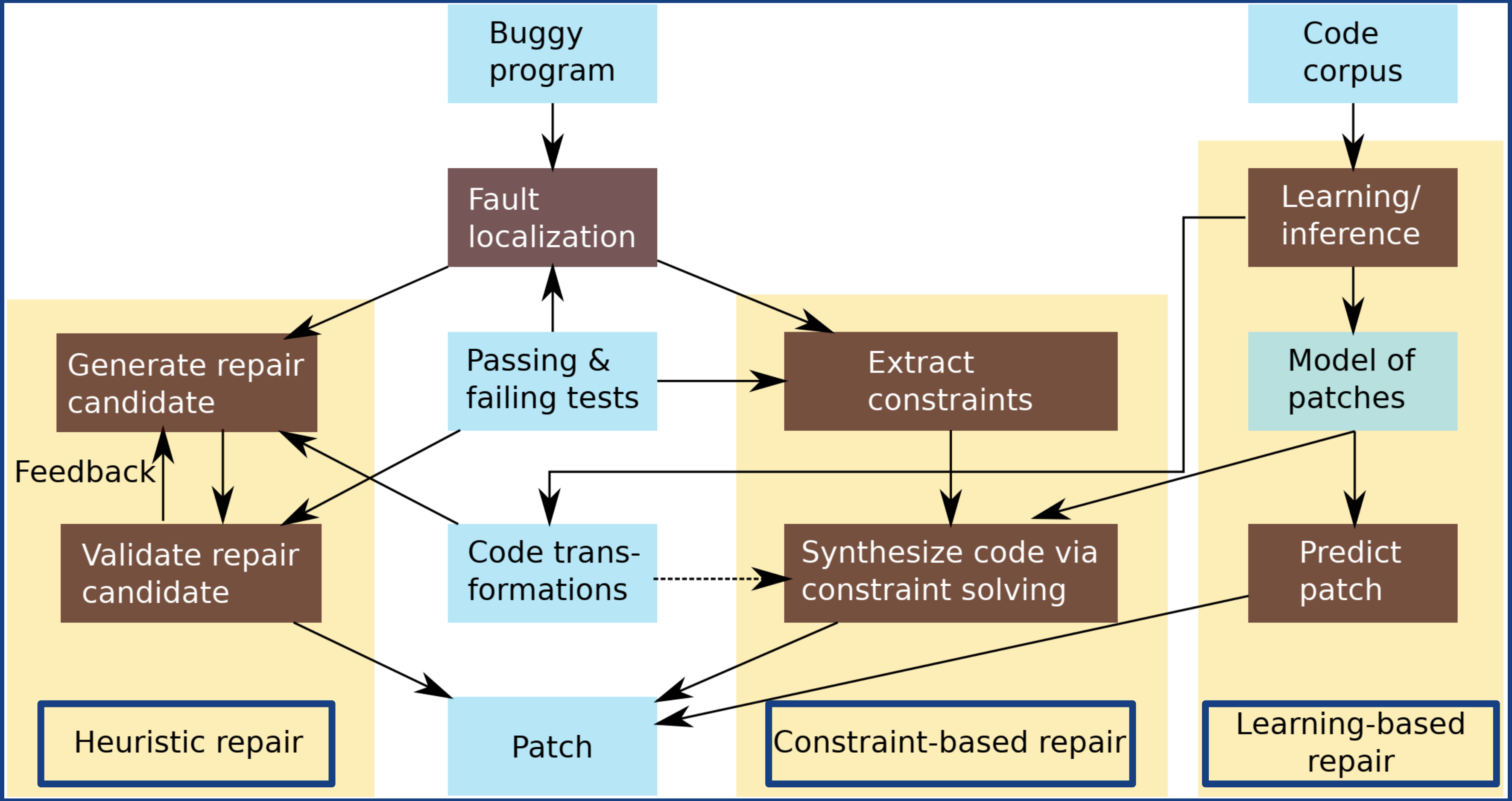


Figure from:
Automated Program Repair. *Claire Le Goues, Michael Pradel, Abhik Roychoudhury.* Communications of the ACM, 2019

Automated Program Repair (APR)

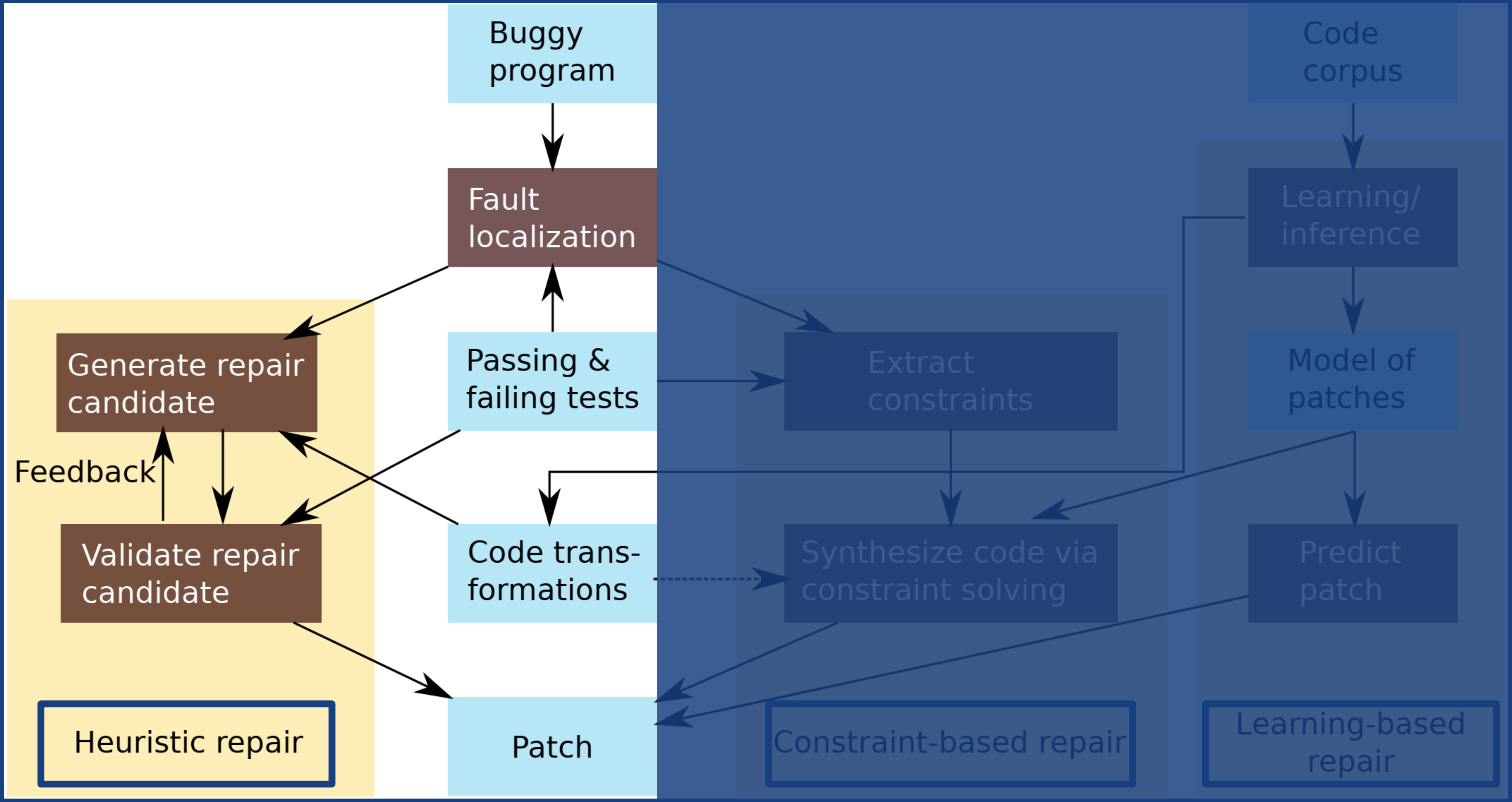


Figure from:
Automated Program Repair. *Claire Le Goues, Michael Pradel, Abhik Roychoudhury.* Communications of the ACM, 2019

Automated Program Repair (APR)

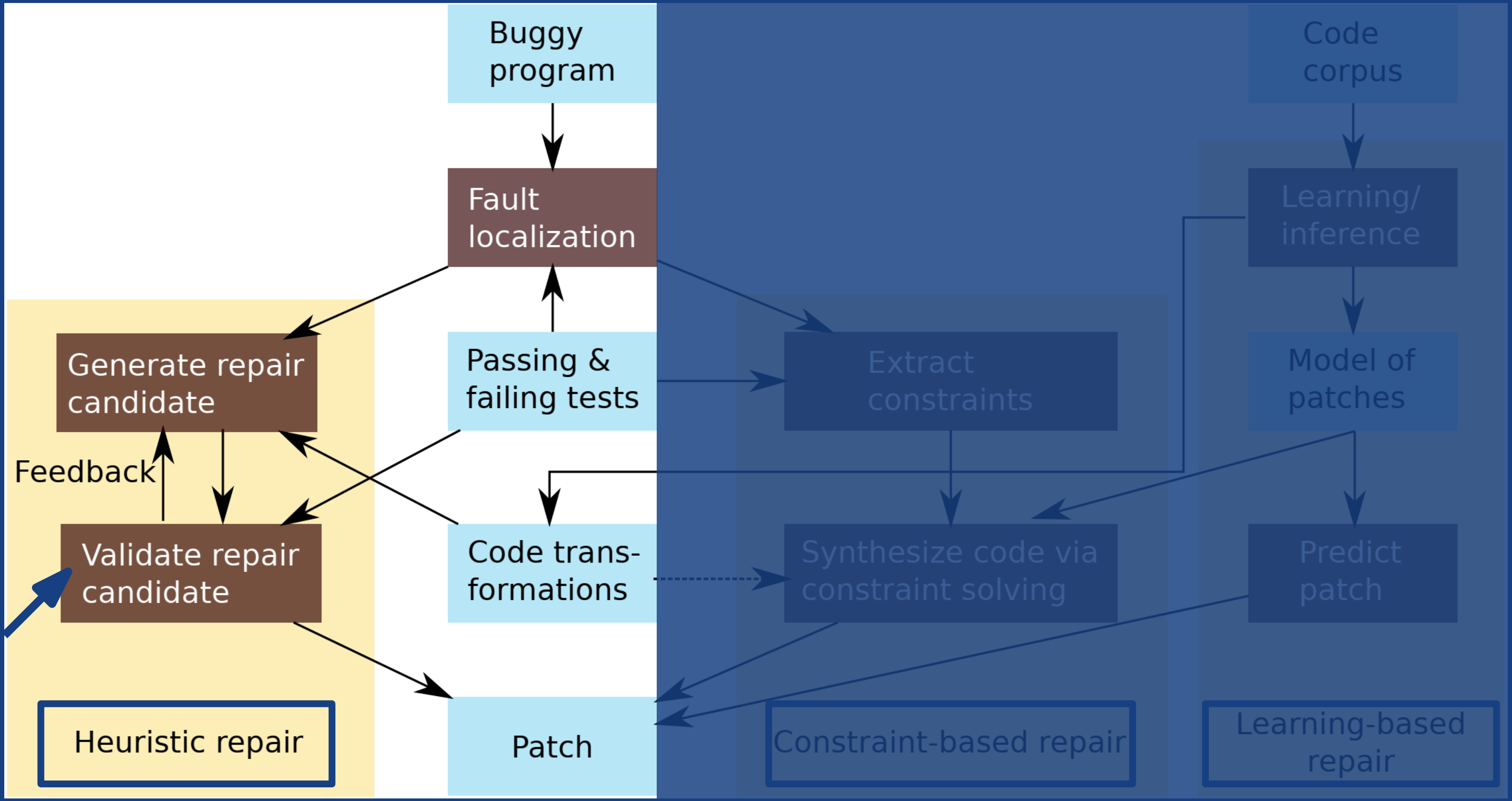


Figure from:
Automated Program Repair. *Claire Le Goues, Michael Pradel, Abhik Roychoudhury.* Communications of the ACM, 2019

Patch Validation in APR

Search-based APR yields many plausible patches

- Test suites are weak oracles
- APR patches overfit to test suites
- Patches pass test suites but fail in practice!

Patch Validation in APR

Search-based APR yields many plausible patches

- Test suites are weak oracles
- APR patches overfit to test suites
- Patches pass test suites but fail in practice!

Existing solutions

- More (& better) tests
- Post-processing (e.g. select smaller patches, use ML on code features, test-based heuristics, . . .)

Automated Program Repair (APR)

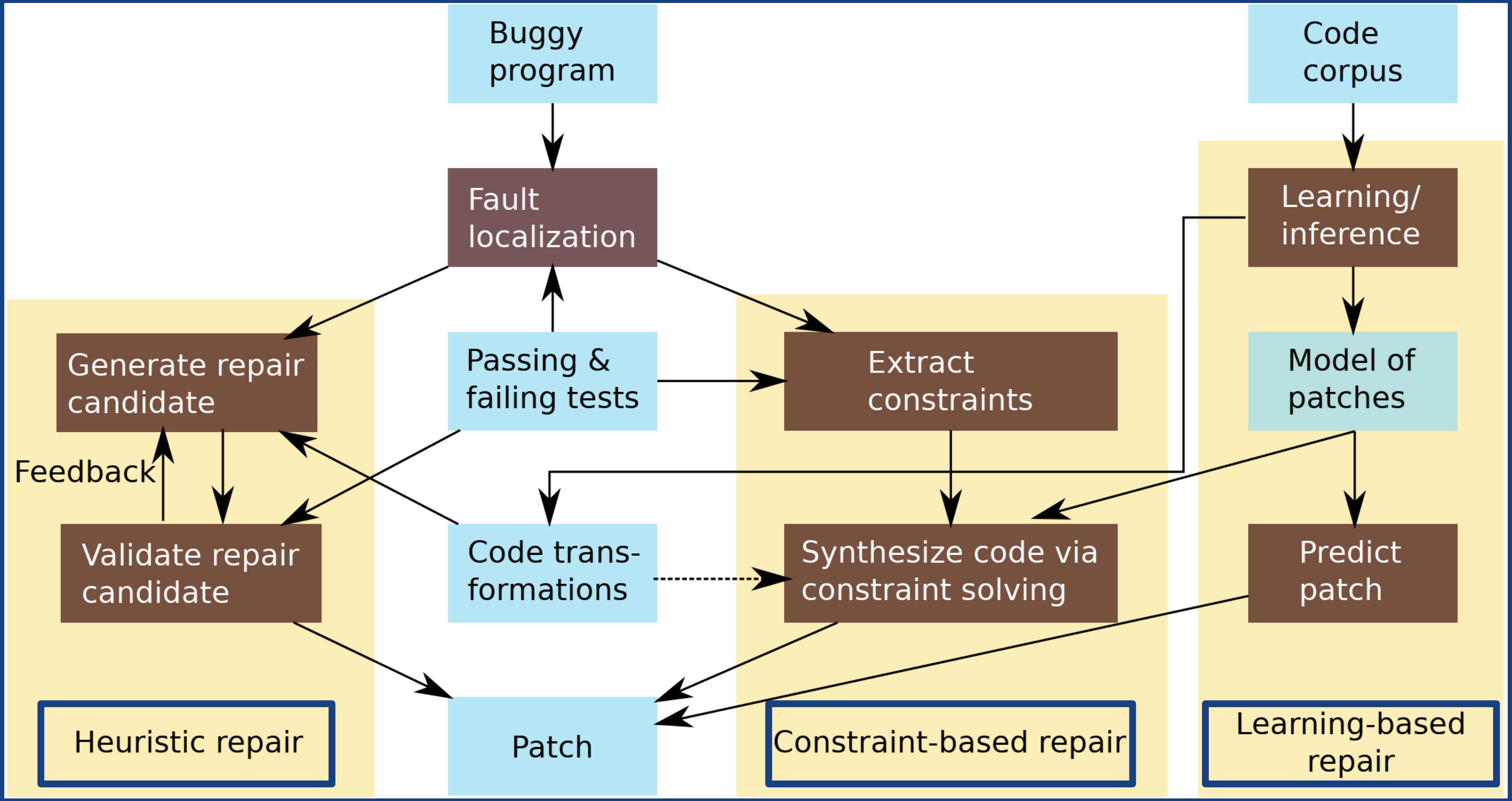


Figure from:
Automated Program Repair. *Claire Le Goues, Michael Pradel, Abhik Roychoudhury.* Communications of the ACM, 2019

Automated Program Repair (APR)

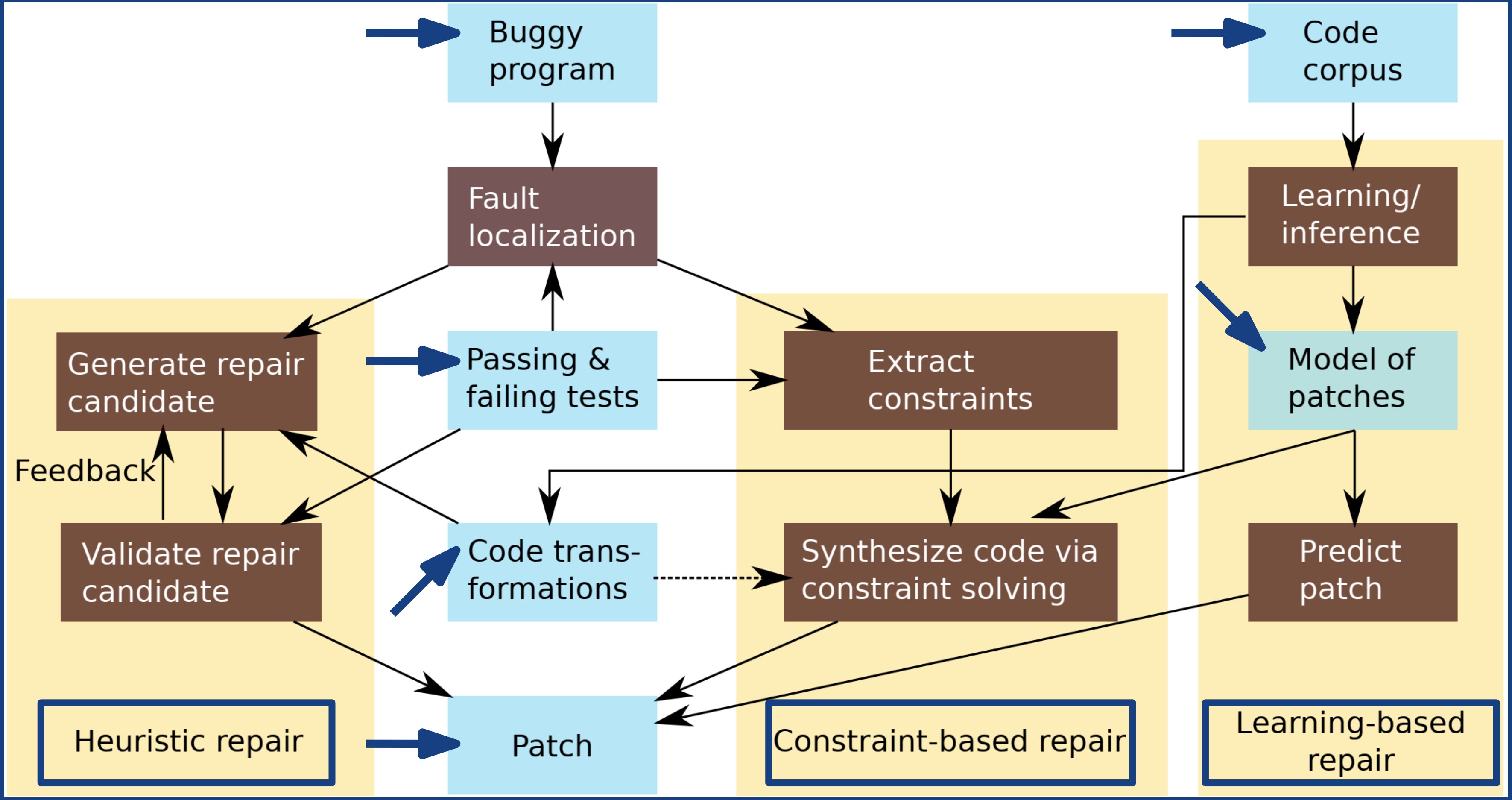
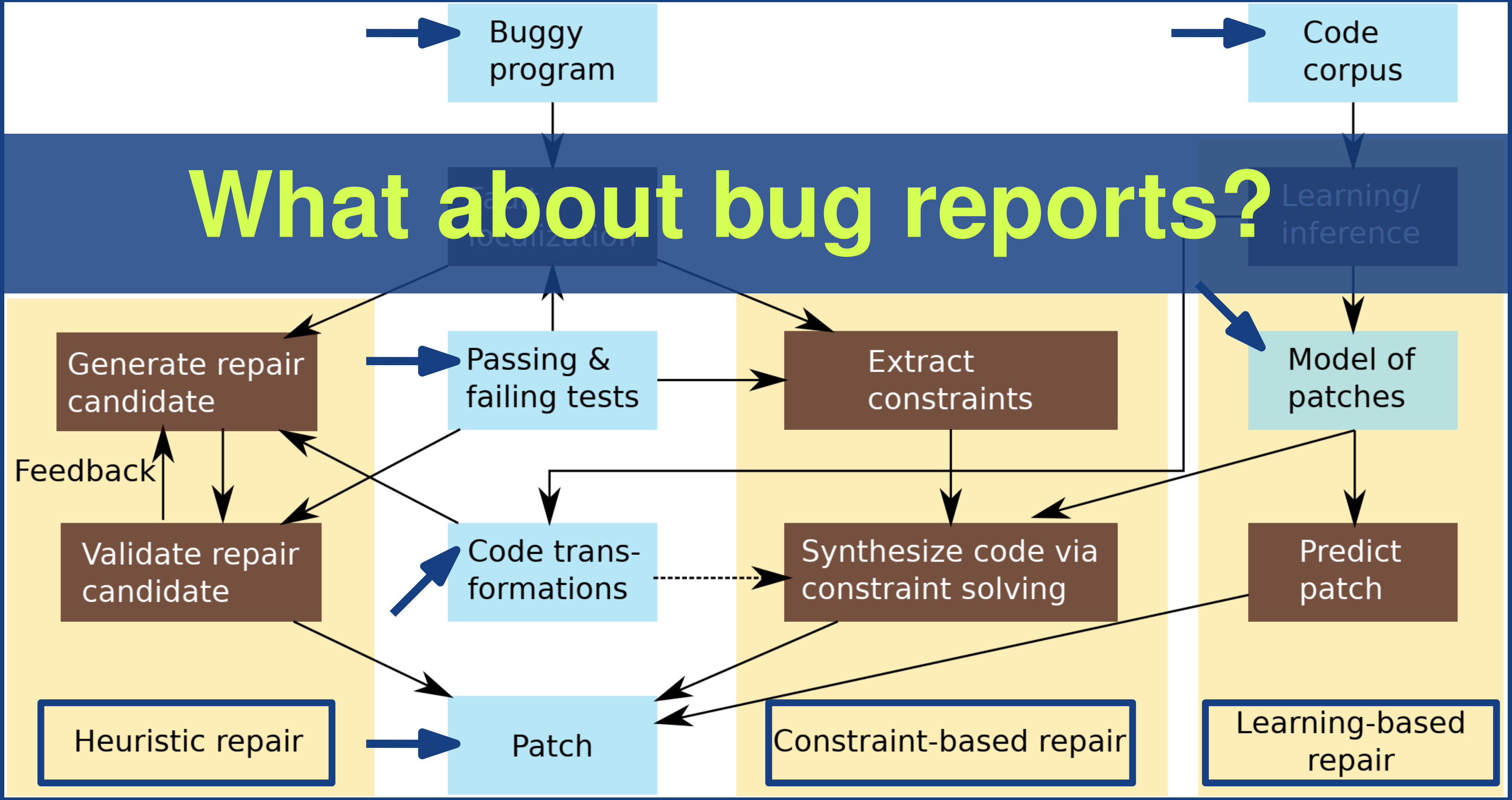


Figure from:
Automated Program Repair. *Claire Le Goues, Michael Pradel, Abhik Roychoudhury.* Communications of the ACM, 2019

Automated Program Repair (APR)



What about bug reports?

Buggy program

Code corpus

Fault localization

Learning/ inference

Generate repair candidate

Passing & failing tests

Extract constraints

Model of patches

Feedback

Validate repair candidate

Code trans- formations

Synthesize code via constraint solving

Predict patch

Heuristic repair

Patch

Constraint-based repair

Learning-based repair

Figure from:
Automated Program Repair. *Claire Le Goues, Michael Pradel, Abhik Roychoudhury.* Communications of the ACM, 2019

Bug Reports (BR) in APR

Bug Reports (BR) in APR

- In the fault localization (FL) component of APR [1, 2]

[1] **iFixR: Bug Report driven Program Repair.** *Koyuncu et. al.* FSE 2019

[2] **Automatically Repairing Programs Using Both Tests and Bug Reports** *Manish Motwani and Yuriy Brun.* arXiv 2022

Bug Reports (BR) in APR

- In the fault localization (FL) component of APR [1, 2]

- For bug classification to select a suitable fix pattern for APR [3]

[1] **iFixR: Bug Report driven Program Repair.** *Koyuncu et. al.* FSE 2019

[2] **Automatically Repairing Programs Using Both Tests and Bug Reports** *Manish Motwani and Yuriy Brun.* arXiv 2022

[3] **R2Fix: Automatically Generating Bug Fixes from Bug Reports.** *Liu et. al.* ICST 2013

Bug Reports (BR) in APR

- In the fault localization (FL) component of APR [1, 2]

- For bug classification to select a suitable fix pattern for APR [3]

- That's it!

[1] **iFixR: Bug Report driven Program Repair.** *Koyuncu et. al.* FSE 2019

[2] **Automatically Repairing Programs Using Both Tests and Bug Reports** *Manish Motwani and Yuriy Brun.* arXiv 2022

[3] **R2Fix: Automatically Generating Bug Fixes from Bug Reports.** *Liu et. al.* ICST 2013

Bug Reports & Patches

Bug Reports & Patches

How to exploit the relation between a bug report and its fixing patch?

Bug Reports & Patches

How to exploit the relation between a bug report and its fixing patch?

Developers write patches in response to bug reports.

Patch Validation as QA

Bug report **Patch**

Patch Validation as QA

Bug report

Describes the problem

I.e. the **Question**

Patch

Patch Validation as QA

Bug report

Describes the problem

I.e. the **Question**

Patch

Describes the solution

I.e. the **Answer**

Patch Validation as QA

Bug report

Describes the problem

I.e. the **Question**

Mostly in NL, (with some code, stack trace, ...)

Patch

Describes the solution

I.e. the **Answer**

Patch Validation as QA

Bug report

Describes the problem

I.e. the **Question**

Mostly in NL, (with some code, stack trace, ...)

Patch

Describes the solution

I.e. the **Answer**

Code diff

Patch Validation as QA

Bug report

Describes the problem

I.e. the **Question**

Mostly in NL, (with some code, stack trace, ...)

How?

Patch

Describes the solution

I.e. the **Answer**

Code diff

Patch Validation as QA

Bug report

↓

Describes the problem

I.e. the **Question**

↓

Mostly in NL, (with some code, stack trace, ...)

← **How?** →

Patch

↓

Describes the solution

I.e. the **Answer**

↓

Code diff ✗

↓

NL description ✓ (e.g. commit msg)

Patch Validation as QA

Bug report

↓

Describes the problem

I.e. the **Question**

↓

Mostly in NL,
(with some code,
stack trace, ...)

Patch

↓

Describes the solution

I.e. the **Answer**

↓

Code diff ✗

↓

NL description ✓
(e.g. commit msg)

← **NLP models** →

Example

Bug report (Closure-96, Defects4J):

“Missing type-checks for `var_args` notation”

Example

Bug report (Closure-96, Defects4J):

“Missing type-checks for `var_args` notation”

Developer patch:

```
     while (arguments.hasNext() &&
-           parameters.hasNext()) {
+           (parameters.hasNext() ||
+            parameter != null && parameter.isVarArgs())) {
       // If there are no parameters left in the list, then the
     while loop
       // above implies that this must be a var_args function.
+      if (parameters.hasNext()) {
         parameter = parameters.next();
+      }
```

Example

Bug report (Closure-96, Defects4J):

“Missing type-checks for `var_args` notation”

Developer patch:

```
     while (arguments.hasNext() &&
-           parameters.hasNext()) {
+           (parameters.hasNext() ||
+            parameter != null && parameter.isVarArgs())) {
       // If there are no parameters left in the list, then the
     while loop
       // above implies that this must be a var_args function.
+      if (parameters.hasNext()) {
         parameter = parameters.next();
+      }
```

Developer commit message for patch:

“check `var_args` properly”

Example

Bug report (Closure-96, Defects4J):

“Missing type-checks for `var_args` notation”

Developer patch:

```
     while (arguments.hasNext() &&
-            parameters.hasNext()) {
+             (parameters.hasNext() ||
+              parameter != null && parameter.isVarArgs())) {
       // If there are no parameters left in the list, then the
     while loop
       // above implies that this must be a var_args function.
+      if (parameters.hasNext()) {
         parameter = parameters.next();
+      }
```

Developer commit message for patch:

“check `var_args` properly”

Example

Bug report (Closure-96, Defects4J):

“Missing type-checks for `var_args` notation”

Developer patch:

```
     while (arguments.hasNext() &&
-           parameters.hasNext()) {
+           (parameters.hasNext() ||
+            parameter != null && parameter.isVarArgs())) {
       // If there are no parameters left in the list, then the
     while loop
       // above implies that this must be a var_args function.
+      if (parameters.hasNext()) {
         parameter = parameters.next();
+      }
```

Developer commit message for patch:

“check `var_args` properly”

Bug Reports & Patches (2)

Semantic relation between BRs and NL patch descriptions?

- Collect BRs and commit messages from Defects4J
- Ground truth: original pairs of (BR, Developer commit message)
- Pairs of not matching (BR, Unrelated developer commit message)
- Vectorize using BERT, and measure Eucleadean distance between the BR and the commit message

Bug Reports & Patches (2)

Semantic relation between BRs and NL patch descriptions?

- Collect BRs and commit messages from Defects4J
- Ground truth: original pairs of (BR, Developer commit message)
- Pairs of not matching (BR, Unrelated developer commit message)
- Vectorize using BERT, and measure Eucleadean distance between the BR and the commit message

Bug Reports & Patches (2)

Semantic relation between BRs and NL patch descriptions?

- Collect BRs and commit messages from Defects4J
- Ground truth: original pairs of (BR, Developer commit message)
- Pairs of not matching (BR, Unrelated developer commit message)
- Vectorize using BERT, and measure Eucleadean distance between the BR and the commit message

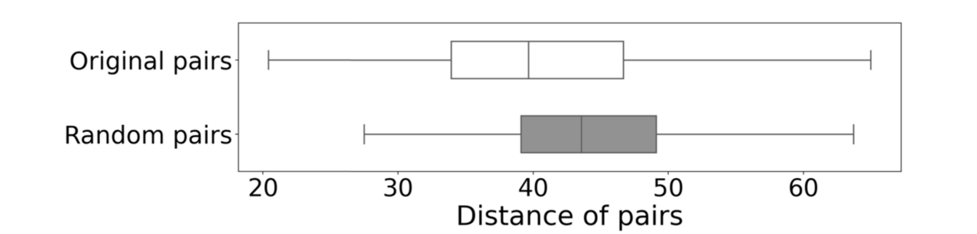


Figure 2: Distributions of Euclidean distances between bug and patch descriptions.

Bug Reports & Patches (2)

Semantic relation between BRs and NL patch descriptions?

- Collect BRs and commit messages from Defects4J
- Ground truth: original pairs of (BR, Developer commit message)
- Pairs of not matching (BR, Unrelated developer commit message)
- Vectorize using BERT, and measure Eucleadean distance between the BR and the commit message
- Mann–Whitney U-test: p-value of 1.2e-32

Bug Reports & Patches (2)

Semantic relation between BRs and NL patch descriptions?

- Collect BRs and commit messages from Defects4J
- Ground truth: original pairs of (BR, Developer commit message)
- Pairs of not matching (BR, Unrelated developer commit message)
- Vectorize using BERT, and measure Eucleadean distance between the BR and the commit message
- Mann–Whitney U-test: p-value of 1.2e-32

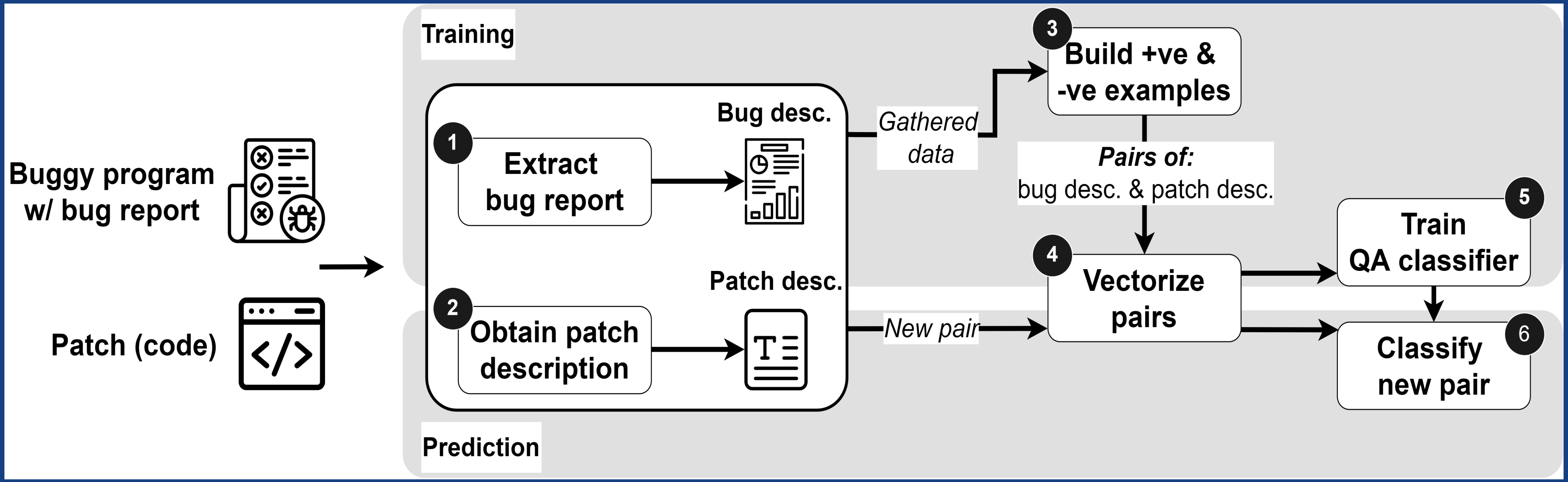
The two distributions are different!

Overview of Quatrain

Question-answering for patch correctness evaluation

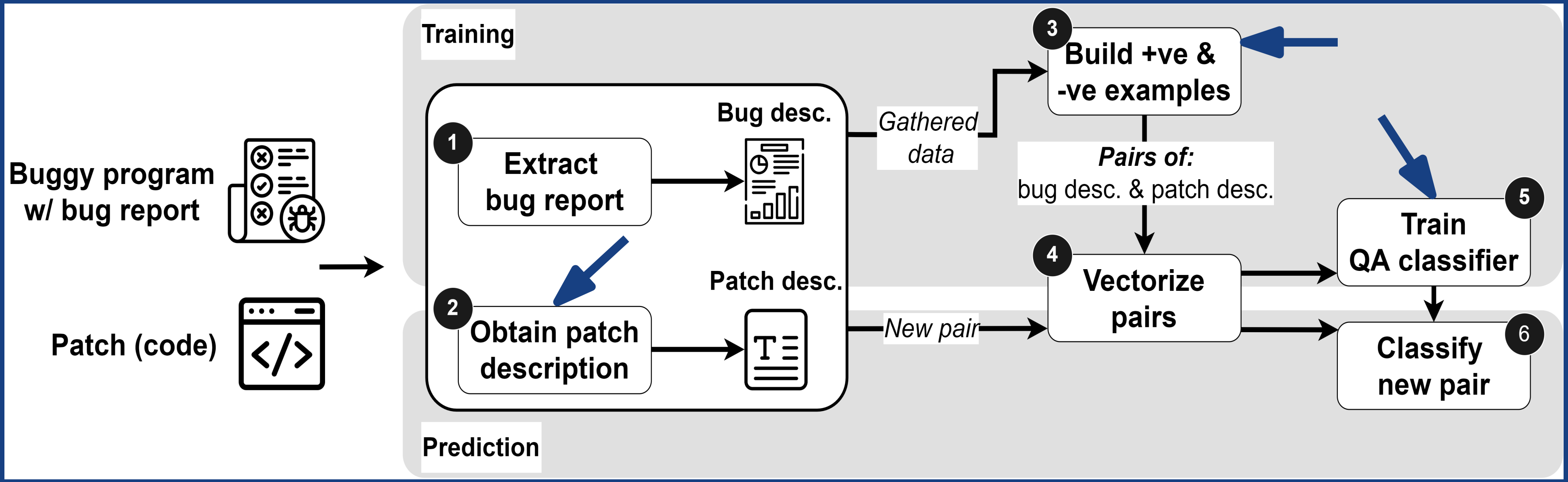
Overview of Quatrain

Question-answering for patch correctness evaluation



Overview of Quatrain

Question-answering for patch correctness evaluation



Obtaining Patch Description

Two cases:

Obtaining Patch Description

Two cases:

- **Developer-written patch:**

  $\Rightarrow$ Use developer-provided commit message

Obtaining Patch Description

Two cases:

- **Developer-written patch:**
  ⇒ Use developer-provided commit message

- **APR-generated patch:**
  ⇒ Generate a patch summary using SOTA code-change summarization

Building Training Examples

Pairs of (Bug report, patch description)

Building Training Examples

Pairs of (Bug report, patch description)

Positive (correct) examples

- (BR, Developer commit message)
- (BR, APR-patch manually labelled ‘correct’)

Building Training Examples

Pairs of (Bug report, patch description)

Positive (correct) examples

- (BR, Developer commit message)
- (BR, APR-patch manually labelled ‘correct’)

Negative (incorrect) examples

- (BR, **Unrelated** developer commit message)
- (BR, APR-patch manually **labelled ‘incorrect’**)

Train QA Classifier

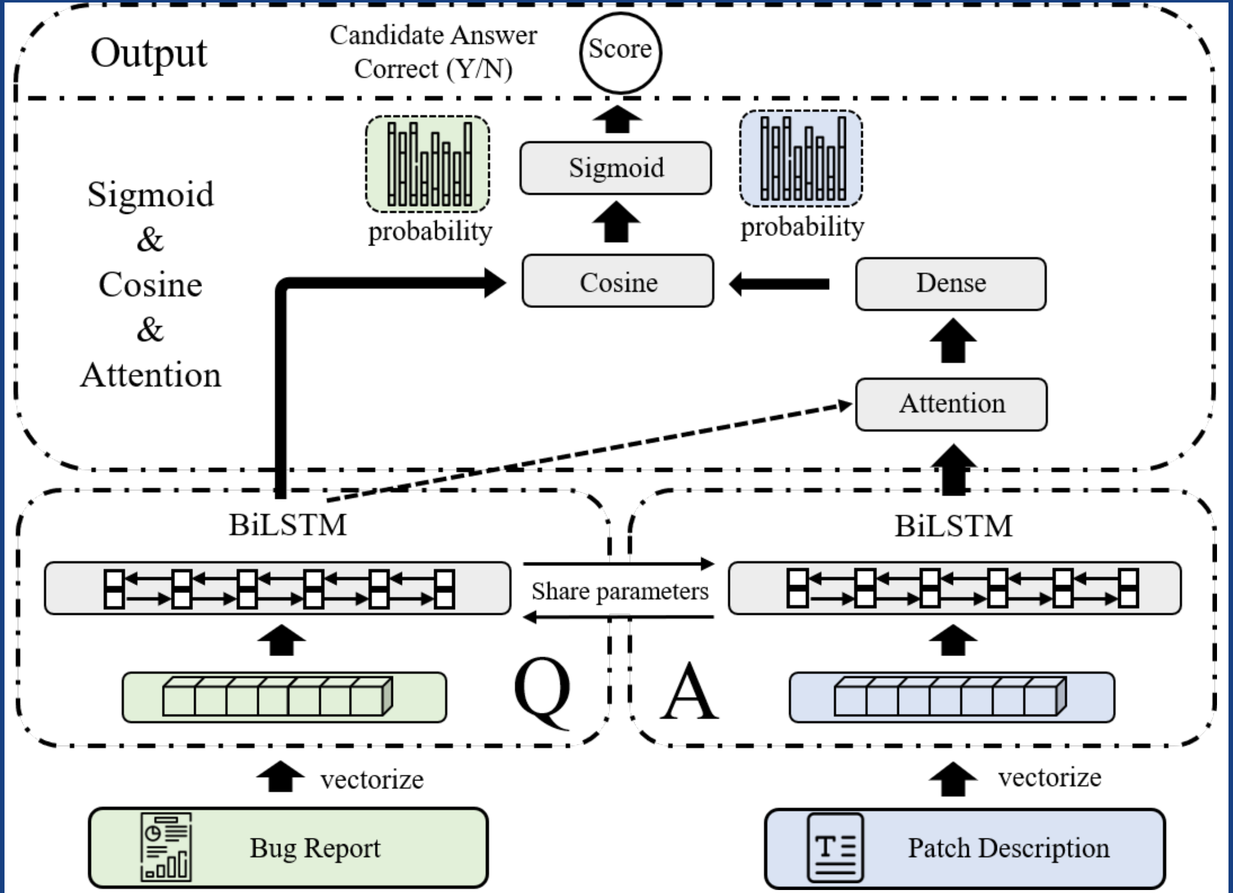
Binary classification: Given a pair of (BR, PatchDesc.), does the patch description answers (solves) the bug report or not.

Learn a function:

f : **(Bug report, patch description)** $\rightarrow \{0, 1\}$

Train QA Classifier

- SOTA QA-model from NLP
- Bi-LSTM with attention



Evaluation: Setup

Dataset

- Defects4J, Bugs.jar, Bears
- Collected bug reports for the associated bugs
- Manually labeled APR-patches from prev. work
- Deduplicate patches
- 9,135 bugs with BRs and labeled patches
  - 1,591 (17.4 %) Correct patches
  - 7,544 (82.6 %) Incorrect patches

Evaluation: Setup

Metrics

- AUC ROC
- F1
- Recall
  - □ +Recall = $\frac{TP}{TP+FN}$
  - □ - Recall = $\frac{TN}{TN+FP}$

Evaluation: Setup

Experimental Setup

- 10-group cross validation
- Split data by bug id, prevent data leakage

Evaluation: Setup

Experimental Setup

- 10-group cross validation
- Split data by bug id, prevent data leakage

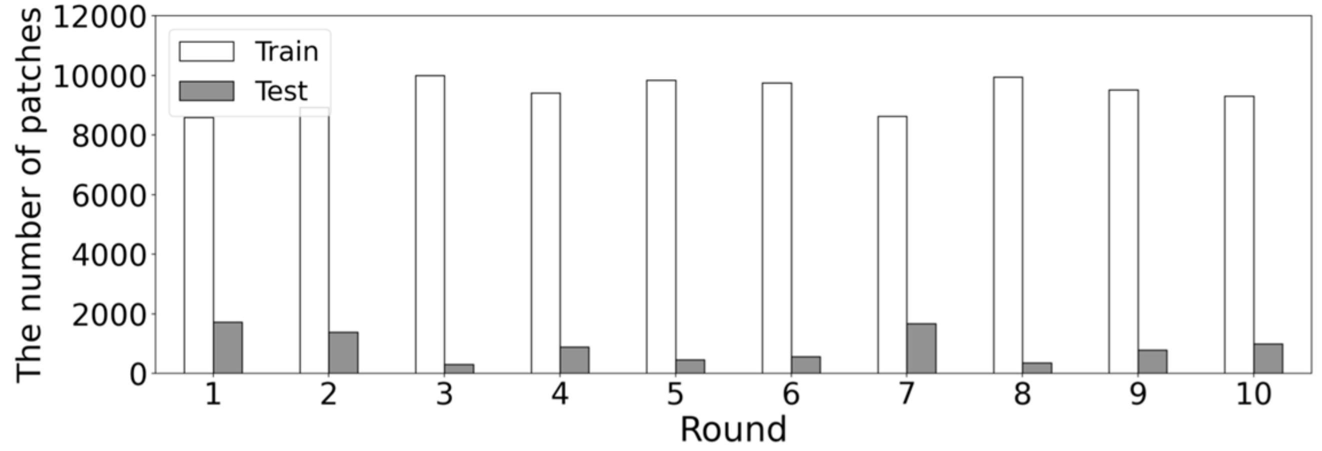


Figure 5: Distribution of Patches in Train and Test Data.

Effectiveness of Quatrain

At a default prediction threshold of 0.5

	AUC	F1 %	+Recall %	-Recall %
Quatrain	0.886	62.8	73.9	87

Effectiveness of Quatrain

At a default prediction threshold of 0.5

	AUC	F1 %	+Recall %	-Recall %
Quatrain	0.886	62.8	73.9	87

Effectiveness of Quatrain

At a default prediction threshold of 0.5

	AUC	F1 %	+Recall %	-Recall %
Quatrain	0.886	62.8	73.9	87

- F1 is impacted by imbalanced data (17.4% correct, 82.6% incorrect)
- When balancing the data, F1 is at 79.3%

Effectiveness of Quatrain (2)

	Thresholds				
	0.3	0.4	0.5	0.6	0.7
#TP	1,551	1,475	1,175	583	189
#TN	3,010	4,653	6,566	7,261	7,522
#FP	4,534	2,891	978	283	22
#FN	40	116	416	1008	1,402
+Recall(%)	97.5	92.7	73.9	36.6	11.9
- Recall(%)	39.9	61.7	87.0	96.2	99.7

Effectiveness of Quatrain (2)

	Thresholds				
	0.3	0.4	0.5	0.6	0.7
#TP	1,551	1,475	1,175	583	189
#TN	3,010	4,653	6,566	7,261	7,522
#FP	4,534	2,891	978	283	22
#FN	40	116	416	1008	1,402
+Recall(%)	97.5	92.7	73.9	36.6	11.9
- Recall(%)	39.9	61.7	87.0	96.2	99.7

Best balance between +Recall and - Recall

Effectiveness of Quatrain (2)

	Thresholds				
	0.3	0.4	0.5	0.6	0.7
#TP	1,551	1,475	1,175	583	189
#TN	3,010	4,653	6,566	7,261	7,522
#FP	4,534	2,891	978	283	22
#FN	40	116	416	1008	1,402
+Recall(%)	97.5	92.7	73.9	36.6	11.9
- Recall(%)	39.9	61.7	87.0	96.2	99.7

Identify most of the correct patches

while sacrificing - Recall

Effectiveness of Quatrain (2)

	Thresholds				
	0.3	0.4	0.5	0.6	0.7
#TP	1,551	1,475	1,175	583	189
#TN	3,010	4,653	6,566	7,261	7,522
#FP	4,534	2,891	978	283	22
#FN	40	116	416	1008	1,402
+Recall(%)	97.5	92.7	73.9	36.6	11.9
- Recall(%)	39.9	61.7	87.0	96.2	99.7

Filter out most of the incorrect patches

while also incorrectly miss many correct ones

Effectiveness of Quatrain (3)

Against a DL-based approach where input is the source code of the generated patches *

Approach	AUC	F1 %	+Recall %	-Recall %
DL using LR	0.719	44.9	83.3	60.5
DL using RF	0.746	47.0	89.4	59.8
Quatrain	0.886	62.8	92.7	61.7

* **Evaluating Representation Learning of Code Changes for Predicting Patch Correctness in Program Repair.** *Tian et. al.* ASE 2020

Effectiveness of Quatrain (3)

Against a DL-based approach where input is the source code of the generated patches [*]

Approach	AUC	F1 %	+Recall %	-Recall %
DL using LR	0.719	44.9	83.3	60.5
DL using RF	0.746	47.0	89.4	59.8
Quatrain	0.886	62.8	92.7	61.7

[*] **Evaluating Representation Learning of Code Changes for Predicting Patch Correctness in Program Repair.** *Tian et. al.* ASE 2020

Effectiveness of Quatrain (3)

Against PATCH-SIM, an execution-based approach [*]

Approach	AUC	F1	+Recall	-Recall
		%	%	%
PATCH-SIM	0.581	5.3	76.9	39.2
Quatrain	0.792	12.7	76.9	66.7

[*] **Identifying Patch Correctness in Test-Based Program Repair.** *Xiong et. al.* ICSE 2018

Effectiveness of Quatrain (3)

Against PATCH-SIM, an execution-based approach [*]

Approach	AUC	F1 %	+Recall %	-Recall %
PATCH-SIM	0.581	5.3	76.9	39.2
Quatrain	0.792	12.7	76.9	66.7

[*] **Identifying Patch Correctness in Test-Based Program Repair.** *Xiong et. al.* ICSE 2018

Limitations

- Supervised approach, requires labelled data

- Relies on the availability and quality of patch descriptions
  - Would benefit from improvements in code-change summarization

Summary

Patch validation as QA-problem using bug reports

- Bug report is the question & patch is the answer
- Bug reports in APR beyond fault localization
- Code and data available at

  `github.com/Trustworthy-Software/Quatrain`