PatchID: A Overfitting Patches Identification Method for Automated Program Repair

Xuan Zhou[1,2], Xingqi Wang[1,2*†], Dan Wei[1,2†] and Yanli Shao[1,2†]

[1*]School of Computer Science, Hangzhou Dianzi University, Street, Hangzhou, 310018, State, China.
[2]Department, Organization, Street, City, 10587, State, Country.
[3]Department, Organization, Street, City, 610101, State, Country.

*Corresponding author(s). E-mail(s): iiauthor@gmail.com;
Contributing authors: 1978881290@qq.com; iiauthor@gmail.com;
iiauthor@gmail.com;
†These authors contributed equally to this work.

Abstract

The patch generated by automatic program repair (APR) needs to be verified. APR usually uses the test suite as the criterion for verifying the correctness of the patch. However, the test suite is unable to fully represent the oracle of the program, which causes APR to generat a large number of overfitting patches that not only fail to fix the original error, but also cause new errors. In order to reduce the number of overfitting patches generated by APR, this paper proposes an overfitting patches identification method for APR, PatchID. The core idea of PatchID is that the dynamic behavior of the passing tests between the buggy program and the correct patch is the same, however the dynamic behavior of the failing tests between themis different. The algorithm first constructs the dynamic behavior expressions that cause the bug from the program and the test suite, then generates new tests to enhance the original test suite, and finally gets the same dynamic behavior expressions from the patch, and identifies whether the patch is overfitting according to whether the value of the dynamic behavior expression changes with the use of the patch. The paper is evaluated on two datasets consisting of 157 Defects4J patches and 380 Java + JML patches, respectively. PatchID successfully identifies 63 overfitting patches and 15 correct patches on the first dataset, and 169 overfitting

patches on the second dataset. In addition, PatchID classifies overfitting patches into three types of patches. Experiments show that the method proposed in this paper is superior to the existing similar methods.

Keywords: keyword1, Keyword2, Keyword3, Keyword4

1 Introduction

Automatic program repair (APR) has been widely studied in the past decade, and a large number of repair technologies have been proposed. Especially after the emergence of GenProg[1], the APR based on test suite occupies the vast majority. The test suite based APR takes a given test suite as oracle and generates patches that are considered correct if they passes the test suite. Accurally, the test suite is weak and can not fully represent the oracle of the program, which results in the existence of patches that pass all the tests but are still wrong, that is, overfitting patches. As a result, APR technology produces a large number of invalid patches. Current APR technologies are far from mature, and most of them will simply accept patches that pass the test suite. According to [5], the majority of patches generated by GenProg, AE and RSRepair are incorrect. More recent techniques try to find many other strategies (e.g., using human-written patches, repair templates and condition synthesis, bug-fixing instances and forbidden modifications) for repair. However their repair performance is still lower.

Due to the low performance of current APR techniques, software developers have to verify a large amount of patches manually to filter out overfitting patches, which consumes too many resources. Therefore, it has become an urgent problem to identify overfitting patches. If a patch passes the test suite and is considered the correct patch, then simply enhancing the test suite can reduce the number of overfitting patches. However, automatic test generation tools can only generate test inputs, and the appropriate test outputs still need to be determined manually. Even so, these approaches still fail to express a complete oracle. It is very difficult to have a complete oracle for a program, especially for large projects. Nilizadeh[7] used a formal method (specification and verification), JML as correctness criterion to successfully describe a small project. However facing the large project, such as Defects4J[48], JML fails. And it seems impossible to use JML to describe such a large project.

At present, it is already very helpful to identify whether a patch is overfitting or not because quickly identifying overfitting patches can improve the success rate of APR technology. Furthermore, if there is technology to subdivide overfitting patches, it can speed bugs fixing. According to reference [4], overfitting patches can be divided into the following three categories:

- A-Overfitting Patch: The patch does not completely fix the incorrect behavior nor does it destroy the original correct behavior.

- B-Overfitting Patch: The patch that fixes the original incorrect behavior but destroys the original correct behavior, which is also called regression.
- AB-Overfitting Patch: The patch destroys the original correct behavior instead of fixing the incorrect behavior.

Because the test suite is not complete, it is far from sufficient to use the consistency of the actual output of the program and the test output, that is to say, the current APR technology does not take full advantage of the test suite, only uses the input and the output of the test suite, and does not dig into the hidden information in the test suite. At present, different overfitting identification methods have been proposed. Among them, the strategy proposed by Xiong [8] is the first technology to mine the deep behavior of test suites and programs. Through the ideas of TEST-SIM and PATCH-SIM, the identification rate of overfitting patch can reach 56

The goal of PatchID is to determine the category of a patch by digging into the correct behavior of the program based on a given test suite. While patches are able to pass the test suite, the passing tests reflect the correct behavior of the program, and the failing test reflect the wrong behavior of the program. From this point of view, patches should maintain the correct behavior of the program and modify the wrong behavior of the program. We believe that there are specific relationships between variables in a correct program, which are reflected by their values at runtime, and these relationships are a manifestation of the correct behavior of the program. So PatchID identifies a patch based on the following two important observations, which are from the perspective of program runtime variables:

To sum up, the contributions of this paper are:

• 5-tuple for computing program execution similarity is proposed. • An overfitting patch identification and subdivision algorithm is designed for automatic patch generation. • The framework of overfitting patch identification is implemented and the proposed method is evaluated. The results show that the proposed method is effective.

The rest of the paper is organized as follows. Section 2 begins with a discussion of related work. Section 3 describes some of the necessary concepts involved in the approach of this paper. Section 4 describes the method in this paper in detail. Section 5 presents an experimental evaluation of the method presented in this paper. Finally, the paper is summarized in Section 6.

2 Results

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

3 This is an example for first level head—section head

3.1 This is an example for second level head—subsection head

3.1.1 This is an example for third level head—subsubsection head

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

4 Equations

Equations in LaTeX can either be inline or on-a-line by itself ("display equations"). For inline equations use the `$...$` commands. E.g.: The equation $H\psi = E\psi$ is written via the command `$H \psi = E \psi$`.

For display equations (with auto generated equation numbers) one can use the equation or align environments:

$$\|\tilde{X}(k)\|^2 \leq \frac{\sum\limits_{i=1}^{p}\left\|\tilde{Y}_i(k)\right\|^2 + \sum\limits_{j=1}^{q}\left\|\tilde{Z}_j(k)\right\|^2}{p+q}. \tag{1}$$

where,

$$\begin{aligned} D_\mu &= \partial_\mu - ig\frac{\lambda^a}{2}A^a_\mu \\ F^a_{\mu\nu} &= \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + gf^{abc}A^b_\mu A^a_\nu \end{aligned} \tag{2}$$

Notice the use of `\nonumber` in the align environment at the end of each line, except the last, so as not to produce equation numbers on lines where no equation numbers are required. The `\label{}` command should only be used at the last line of an align environment where `\nonumber` is not used.

$$Y_\infty = \left(\frac{m}{\text{GeV}}\right)^{-3}\left[1 + \frac{3\ln(m/\text{GeV})}{15} + \frac{\ln(c_2/5)}{15}\right] \tag{3}$$

The class file also supports the use of `\mathbb{}`, `\mathscr{}` and `\mathcal{}` commands. As such `\mathbb{R}`, `\mathscr{R}` and `\mathcal{R}` produces $\mathbb{R}$, $\mathscr{R}$ and $\mathcal{R}$ respectively (refer Subsubsection 3.1.1).

5 Tables

Tables can be inserted via the normal table and tabular environment. To put footnotes inside tables you should use `\footnotetext[]{...}` tag. The

footnote appears just below the table itself (refer Tables 1 and 2). For the corresponding footnotemark use `\footnotemark[...]`

Table 1 Caption text

Column 1	Column 2	Column 3	Column 4
row 1	data 1	data 2	data 3
row 2	data 4	data 5[1]	data 6
row 3	data 7	data 8	data 9[2]

Source: This is an example of table footnote. This is an example of table footnote.

[1]Example for a first table footnote. This is an example of table footnote.

[2]Example for a second table footnote. This is an example of table footnote.

The input format for the above table is as follows:

```
\begin{table}[<placement-specifier>]
\begin{center}
\begin{minipage}{<preferred-table-width>}
\caption{<table-caption>}\label{<table-label>}%
\begin{tabular}{@{}llll@{}}
\toprule
Column 1 & Column 2 & Column 3 & Column 4\\
\midrule
row 1 & data 1 & data 2  & data 3 \\
row 2 & data 4 & data 5\footnotemark[1] & data 6 \\
row 3 & data 7 & data 8  & data 9\footnotemark[2]\\
\botrule
\end{tabular}
\footnotetext{Source: This is an example of table footnote.
This is an example of table footnote.}
\footnotetext[1]{Example for a first table footnote.
This is an example of table footnote.}
\footnotetext[2]{Example for a second table footnote.
This is an example of table footnote.}
\end{minipage}
\end{center}
\end{table}
```

In case of double column layout, tables which do not fit in single column width should be set to full text width. For this, you need to use `\begin{table*} ... \end{table*}` instead of `\begin{table} ... \end{table}` environment. Lengthy tables which do not fit in textwidth should

Table 2 Example of a lengthy table which is set to full textwidth

	Element 1[1]			Element 2[2]		
Project	Energy	σ_{calc}	σ_{expt}	Energy	σ_{calc}	σ_{expt}
Element 3	990 A	1168	1547 ± 12	780 A	1166	1239 ± 100
Element 4	500 A	961	922 ± 10	900 A	1268	1092 ± 40

Note: This is an example of table footnote. This is an example of table footnote this is an example of table footnote this is an example of table footnote this is an example of table footnote.

[1]Example for a first table footnote.

[2]Example for a second table footnote.

be set as rotated table. For this, you need to use `\begin{sidewaystable} ... \end{sidewaystable}` instead of `\begin{table*} ... \end{table*}` environment. This environment puts tables rotated to single column width. For tables rotated to double column width, use `\begin{sidewaystable*} ... \end{sidewaystable*}`.

6 Figures

As per the LaTeX standards you need to use eps images for LaTeX compilation and `pdf/jpg/png` images for `PDFLaTeX` compilation. This is one of the major difference between LaTeX and `PDFLaTeX`. Each image should be from a single input .eps/vector image file. Avoid using subfigures. The command for inserting images for LaTeX and `PDFLaTeX` can be generalized. The package used to insert images in `LaTeX/PDFLaTeX` is the graphicx package. Figures can be inserted via the normal figure environment as shown in the below example:

```
\begin{figure}[<placement-specifier>]
\centering
\includegraphics{<eps-file>}
\caption{<figure-caption>}\label{<figure-label>}
\end{figure}
```

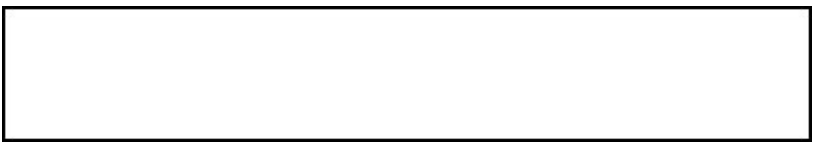

Fig. 1 This is a widefig. This is an example of long caption this is an example of long caption this is an example of long caption this is an example of long caption

In case of double column layout, the above format puts figure captions/images to single column width. To get spanned images, we need to provide `\begin{figure*} ... \end{figure*}`.

Table 3 Tables which are too long to fit, should be written using the "sidewaystable" environment as shown here

	Element 1[1]			Element[2]		
Projectile	Energy	σ_{calc}	σ_{expt}	Energy	σ_{calc}	σ_{expt}
Element 3	990 A	1168	1547 ± 12	780 A	1166	1239 ± 100
Element 4	500 A	961	922 ± 10	900 A	1268	1092 ± 40
Element 5	990 A	1168	1547 ± 12	780 A	1166	1239 ± 100
Element 6	500 A	961	922 ± 10	900 A	1268	1092 ± 40

Note: This is an example of table footnote this is an example of table footnote this is an example of table footnote this is an example of table footnote this is an example of table footnote.

[1]This is an example of table footnote.

For sample purpose, we have included the width of images in the optional argument of `\includegraphics` tag. Please ignore this.

7 Algorithms, Program codes and Listings

Packages `algorithm`, `algorithmicx` and `algpseudocode` are used for setting algorithms in LaTeX using the format:

```
\begin{algorithm}
\caption{<alg-caption>}\label{<alg-label>}
\begin{algorithmic}[1]
. . .
\end{algorithmic}
\end{algorithm}
```

You may refer above listed package documentations for more details before setting `algorithm` environment. For program codes, the "program" package is required and the command to be used is `\begin{program} ... \end{program}`. A fast exponentiation procedure:

```
begin
  for i := 1 to 10 step 1 do
      expt(2, i);
      newline() od                Comments will be set flush to the right margin
where
proc expt(x, n) ≡
  z := 1;
  do if n = 0 then exit fi;
      do if odd(n) then exit fi;
          comment: This is a comment statement;
          n := n/2; x := x * x od;
      {n > 0};
      n := n − 1; z := z * x od;
  print(z).
end
```

Similarly, for `listings`, use the `listings` package. `\begin{lstlisting} ... \end{lstlisting}` is used to set environments similar to `verbatim` environment. Refer to the `lstlisting` package documentation for more details.

Algorithm 1 Calculate $y = x^n$

Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$
1: $y \Leftarrow 1$
2: **if** $n < 0$ **then**
3: $X \Leftarrow 1/x$
4: $N \Leftarrow -n$
5: **else**
6: $X \Leftarrow x$
7: $N \Leftarrow n$
8: **end if**
9: **while** $N \neq 0$ **do**
10: **if** N is even **then**
11: $X \Leftarrow X \times X$
12: $N \Leftarrow N/2$
13: **else**[N is odd]
14: $y \Leftarrow y \times X$
15: $N \Leftarrow N - 1$
16: **end if**
17: **end while**

```
for i:=maxint to 0 do
begin
{ do nothing }
end;
Write('Case insensitive ');
Write('Pascal keywords.');
```

8 Cross referencing

Environments such as figure, table, equation and align can have a label declared via the `\label{#label}` command. For figures and table environments use the `\label{}` command inside or just below the `\caption{}` command. You can then use the `\ref{#label}` command to cross-reference them. As an example, consider the label declared for Figure 1 which is `\label{fig1}`. To cross-reference it, use the command `Figure \ref{fig1}`, for which it comes up as "Figure 1".

To reference line numbers in an algorithm, consider the label declared for the line number 2 of Algorithm 1 is `\label{algln2}`. To cross-reference it, use the command `\ref{algln2}` for which it comes up as line 2 of Algorithm 1.

8.1 Details on reference citations

Standard LaTeX permits only numerical citations. To support both numerical and author-year citations this template uses `natbib` LaTeX package. For style guidance please refer to the template user manual.

Here is an example for `\cite{...}`: [1]. Another example for `\citep{...}`: [2]. For author-year citation mode, `\cite{...}` prints Jones et al. (1990) and `\citep{...}` prints (Jones et al., 1990).

All cited bib entries are printed at the end of this article: [3], [4], [5], [6], [7], [8], [9], [10], [11] and [12].

9 Examples for theorem like environments

For theorem like environments, we require `amsthm` package. There are three types of predefined theorem styles exists—`thmstyleone`, `thmstyletwo` and `thmstylethree`

`thmstyleone`	Numbered, theorem head in bold font and theorem text in italic style
`thmstyletwo`	Numbered, theorem head in roman font and theorem text in italic style
`thmstylethree`	Numbered, theorem head in bold font and theorem text in roman style

For mathematics journals, theorem styles can be included as shown in the following examples:

Theorem 1 (Theorem subhead) *Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text. Example theorem text.*

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Proposition 2 *Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text. Example proposition text.*

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Example 1 Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam

ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem.

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Remark 1 Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem.

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Definition 1 (Definition sub head) Example definition text. Example definition text. Example definition text. Example definition text. Example definition text. Example definition text. Example definition text. Example definition text.

Additionally a predefined "proof" environment is available: `\begin{proof} ... \end{proof}`. This prints a "Proof" head in italic font style and the "body text" in roman font style with an open square at the end of each proof environment.

Proof Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. □

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text. Sample body text.

Proof of Theorem 1 Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. Example for proof text. □

For a quote environment, use `\begin{quote}...\end{quote}`

> Quoted text example. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Sample body text. Sample body text. Sample body text. Sample body text. Sample body text (refer Figure 1). Sample body text. Sample body text. Sample body text (refer Table 3).

10 Methods

Topical subheadings are allowed. Authors must ensure that their Methods section includes adequate experimental and characterization data necessary for others in the field to reproduce their work. Authors are encouraged to include RIIDs where appropriate.

Ethical approval declarations (only required where applicable) Any article reporting experiment/s carried out on (i) live vertebrate (or higher invertebrates), (ii) humans or (iii) human samples must include an unambiguous statement within the methods section that meets the following requirements:

1. Approval: a statement which confirms that all experimental protocols were approved by a named institutional and/or licensing committee. Please identify the approving body in the methods section
2. Accordance: a statement explicitly saying that the methods were carried out in accordance with the relevant guidelines and regulations
3. Informed consent (for experiments involving humans or human tissue samples): include a statement confirming that informed consent was obtained from all participants and/or their legal guardian/s

If your manuscript includes potentially identifying patient/participant information, or if it describes human transplantation research, or if it reports results of a clinical trial then additional information will be required. Please visit (https://www.nature.com/nature-research/editorial-policies) for Nature Portfolio journals, (https://www.springer.com/gp/authors-editors/journal-author/journal-author-helpdesk/publishing-ethics/14214) for Springer Nature journals, or (https://www.biomedcentral.com/getpublished/editorial-policies#ethics+and+consent) for BMC.

11 Discussion

Discussions should be brief and focused. In some disciplines use of Discussion or 'Conclusion' is interchangeable. It is not mandatory to use both. Some journals prefer a section 'Results and Discussion' followed by a section 'Conclusion'. Please refer to Journal-level guidance for any specific requirements.

12 Conclusion

Conclusions may be used to restate your hypothesis or research question, restate your major findings, explain the relevance and the added value of your work, highlight any limitations of your study, describe future directions for research and recommendations.

In some disciplines use of Discussion or 'Conclusion' is interchangeable. It is not mandatory to use both. Please refer to Journal-level guidance for any specific requirements.

Supplementary information. If your article has accompanying supplementary file/s please state so here.

Authors reporting data from electrophoretic gels and blots should supply the full unprocessed scans for key as part of their Supplementary information. This may be requested by the editorial team/s if it is missing.

Please refer to Journal-level guidance for any specific requirements.

Acknowledgments. Acknowledgments are not compulsory. Where included they should be brief. Grant or contribution numbers may be acknowledged.

Please refer to Journal-level guidance for any specific requirements.

Declarations

Some journals require declarations to be submitted in a standardised format. Please check the Instructions for Authors of the journal to which you are submitting to see if you need to complete this section. If yes, your manuscript must contain the following sections under the heading 'Declarations':

- Funding
- Conflict of interest/Competing interests (check journal-specific guidelines for which heading to use)
- Ethics approval
- Consent to participate
- Consent for publication
- Availability of data and materials
- Code availability
- Authors' contributions

If any of the sections are not relevant to your manuscript, please include the heading and write 'Not applicable' for that section.

Editorial Policies for:

Springer journals and proceedings:
https://www.springer.com/gp/editorial-policies

Nature Portfolio journals:
https://www.nature.com/nature-research/editorial-policies

Scientific Reports:
https://www.nature.com/srep/journal-policies/editorial-policies

BMC journals:
https://www.biomedcentral.com/getpublished/editorial-policies

Appendix A Section title of first appendix

An appendix contains supplementary information that is not an essential part of the text itself but which may be helpful in providing a more comprehensive understanding of the research problem or it is information that is too cumbersome to be included in the body of the paper.

References

[1] Campbell, S.L., Gear, C.W.: The index of general nonlinear DAES. Numer. Math. **72**(2), 173–196 (1995)

[2] Slifka, M.K., Whitton, J.L.: Clinical implications of dysregulated cytokine production. J. Mol. Med. **78**, 74–80 (2000). https://doi.org/10.1007/s001090000086

[3] Hamburger, C.: Quasimonotonicity, regularity and duality for nonlinear systems of partial differential equations. Ann. Mat. Pura. Appl. **169**(2), 321–354 (1995)

[4] Geddes, K.O., Czapor, S.R., Labahn, G.: Algorithms for Computer Algebra. Kluwer, Boston (1992)

[5] Broy, M.: Software engineering—from auxiliary to key technologies. In: Broy, M., Denert, E. (eds.) Software Pioneers, pp. 10–13. Springer, New York (1992)

[6] Seymour, R.S. (ed.): Conductive Polymers. Plenum, New York (1981)

[7] Smith, S.E.: Neuromuscular blocking drugs in man. In: Zaimis, E. (ed.) Neuromuscular Junction. Handbook of Experimental Pharmacology, vol. 42, pp. 593–660. Springer, Heidelberg (1976)

[8] Chung, S.T., Morris, R.L.: Isolation and characterization of plasmid deoxyribonucleic acid from Streptomyces fradiae. Paper presented at the 3rd international symposium on the genetics of industrial microorganisms, University of Wisconsin, Madison, 4–9 June 1978 (1978)

[9] Hao, Z., AghaKouchak, A., Nakhjiri, N., Farahmand, A.: Global integrated drought monitoring and prediction system (GIDMaPS) data sets. figshare https://doi.org/10.6084/m9.figshare.853801 (2014)

[10] Babichev, S.A., Ries, J., Lvovsky, A.I.: Quantum scissors: teleportation of single-mode optical states by means of a nonlocal single photon. Preprint at https://arxiv.org/abs/quant-ph/0208066v1 (2002)

[11] Beneke, M., Buchalla, G., Dunietz, I.: Mixing induced CP asymmetries in

inclusive B decays. Phys. Lett. **B393**, 132–142 (1997) https://arxiv.org/abs/0707.3168 [gr-gc]

[12] Stahl, B.: DeepSIP: Deep Learning of Supernova Ia Parameters, 0.42, Astrophysics Source Code Library (2020), https://ascl.net/2006.023