

Shaping Program Repair Space with Existing Patches and Similar Code

jiajun jiang, Yingfei Xiong, Hongyu Zhang, Qing Gao,
Xiangqun Chen

Proceedings of the 2018 ACM SIGSOFT 27th International Symposium on Software Testing and Analysis
(ISSTA) pages 298–309

発表者：辻 裕真

背景

- ・ 自動プログラム修正（APR）はバグ修正の労力を大きく削減できる.
- ・ APRはバグのあるプログラムとテストケースが入力され合格するパッチを生成する.
- ・ テストケースには合格するが正しいとはいえないパッチも生成されることがある.
- ・ 既存のパッチとプログラム内の類似コードからパッチを生成する SimFixを提案する.

APR

- ・ 広大な検索スペース（さまざまなパッチ）の中からテストケースに合格するパッチを検索するプログラムといえる.
- ・ テストケースには合格するが正しいとはいえないパッチを回避する必要がある.
- ・ 多くの手法では検索スペースを制限, 優先順位付けして正しい可能性が高いパッチから試行を行う.
- ・ 検索スペースには既存のパッチや同一プロジェクト内の類似コードが用いられる.

アプローチ

- 提案手法では既存のパッチからなる検索スペースと、同一プロジェクト内の類似コードからなる検索スペースの共通部分からパッチを検索する。

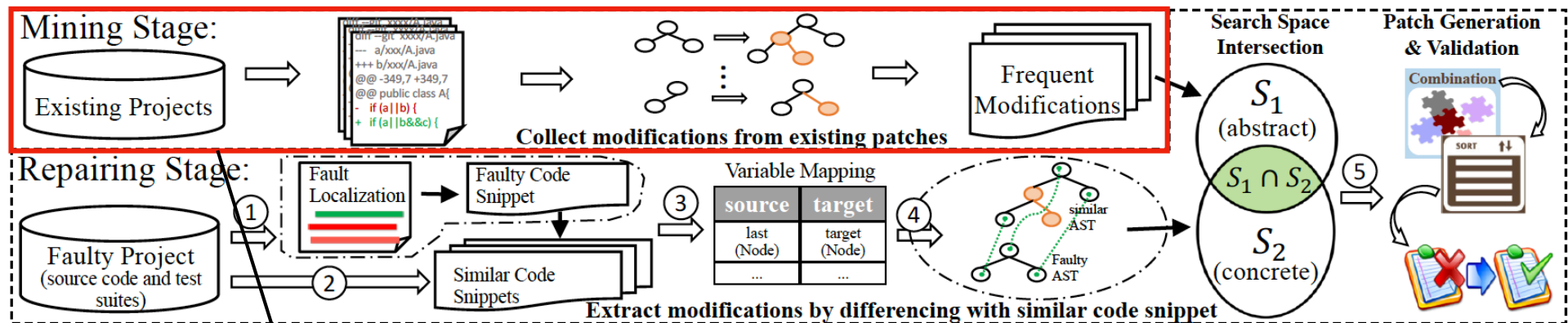


Figure 1: The overall workflow of the proposed approach.

既存のパッチから検索スペースS1を生成

アプローチ

- 提案手法では既存のパッチからなる検索スペースと、同一プロジェクト内の類似コードからなる検索スペースの共通部分からパッチを検索する.

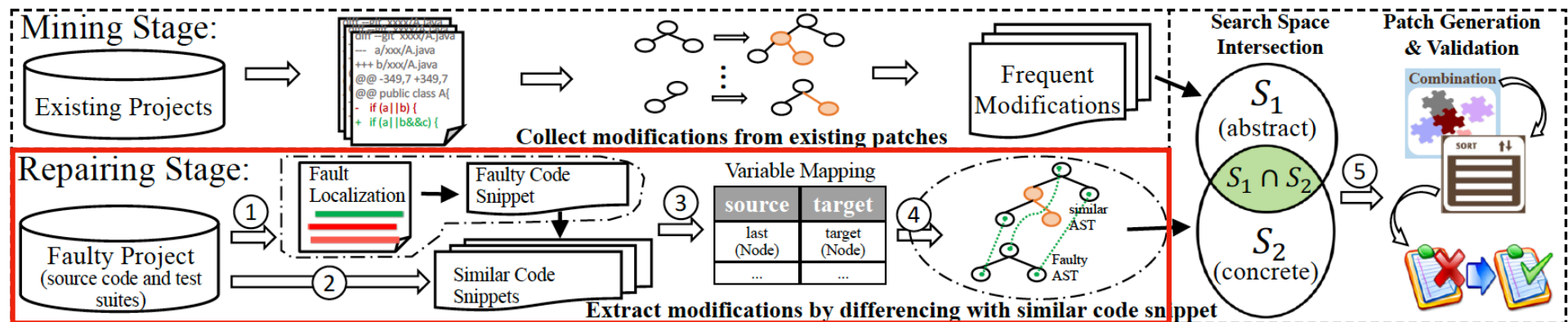


Figure 1: The overall workflow of the proposed approach.

バグの存在するコードと類似するコードを特定し
複数の処理を行い検索スペースS2を生成

アプローチ

- 提案手法では既存のパッチからなる検索スペースと、同一プロジェクト内の類似コードからなる検索スペースの共通部分からパッチを検索する.

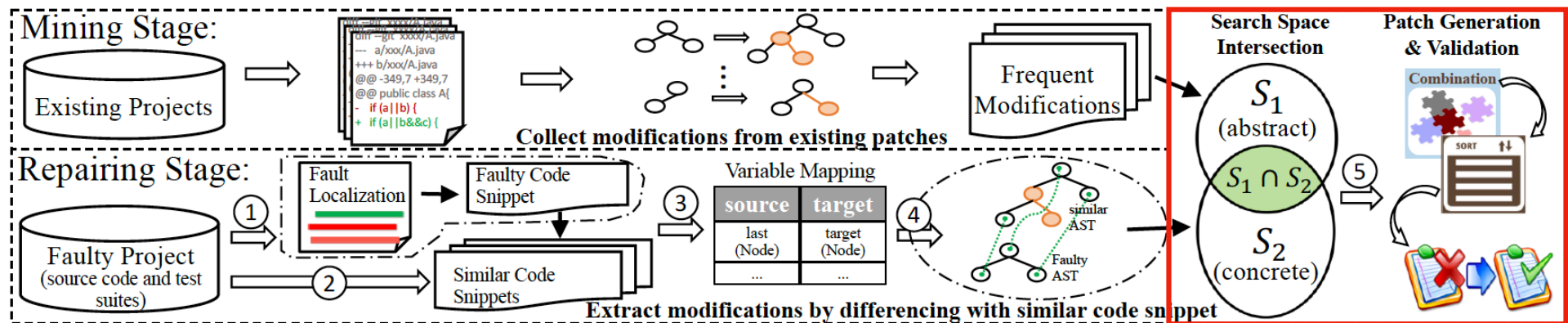


Figure 1: The overall workflow of the proposed approach.

S1とS2の共通部分を検索スペースとして用いパッチを生成

アプローチ

Repairing Stage

- Ochiaiアルゴリズム[1]を使用してバグのある行を特定.
- バグのある行からコードスニペットを定義する.

```
1 if(last != null && last.getType() == Token.STRING){  
2     String propName = last.getString();  
3     return (propName.equals(methodName));  
4 }
```

行を指定しスニペットを定義

1～4行目までの場合：ifステートメント全体

1～2行目までの場合：2行目のみ

バグのある行を中心にスニペットの行数が10行以下になるものを全て抽出

- 同一プロジェクト内から3つの指標（コードの構造, 変数名, メソッド名）から類似度を計算し上位100件の類似コードスニペットを抽出.

[1]Rui Abreu, Peter Zoetewij, and Arjan J. C. van Gemund. 2006.

An Evaluation of Similarity Coefficients for Software Fault Localization (PRDC)

アプローチ

- ・ S1とS2の共通する部分を検索スペースとして、スペース内のパッチをランク付けする.
- ・ 一貫性のあるパッチ
(異なる場所にある同じ変数にも同様の変更を行なっている場合)
- ・ 変更の少ないパッチ
- ・ 挿入よりも置換を行なっているパッチ
- ・ ランクの高いパッチからテストケースに合格するかパッチを試行していく.

評価

修正

- バグデータセットであるDefects4Jを用いて評価を行う.
- 最先端の8つの手法と比較.
- 4つのプロジェクトで最も多くのバグを修正できた.

Table 4: Correct patches generated by different techniques.

Proj.	SimFix	jGP	jKali	Nopol	ACS	HDR	ssFix	ELIXIR	JAID
Chart	4	0	0	1	2	-(2)	3	4	2(4)
Closure	6	0	0	0	0	-(7)	2	0	5(9)
Math	14	5	1	1	12	-(7)	10	12	1(7)
Lang	9	0	0	3	3	-(6)	5	8	1(5)
Time	1	0	0	0	1	-(1)	0	2	0(0)
Total	34	5	1	5	18	13(23)	20	26	9(25)

評価

検索スペースの削減

- SimFixは類似コードから検索スペースを取得し、その中から既存のパッチと一致するものを検索している.
- 既存のパッチとの一致を考慮せず類似コードのみを検索スペースとして用いた場合と比較を行った.
- 修復されたバグの数は12個少なくなり、修正候補は約2.3倍に、所要時間は約2.0倍であった.

まとめ

- ・ 既存のパッチと類似しているコードの2つの修正候補を組み合わせて新しい自動プログラム修正アプローチを提案した.
- ・ 最先端の自動プログラム修正ツールと比較して, 最も多くのバグを修正できた.
- ・ パッチの検索スペースを制限することにより, 正しいパッチが作成できた.

所感

- Dockerfileの自動修正という自分の研究にも活かせそうな内容だった.
- 抽象構文木（AST）を用いた差分の抽出や類似度の計算など，自分でも実装できるように勉強しなければいけない.