유사 패치 기반 자동 프로그램 수정 기법

장세창[1] 최준혁[1] 김성빈[1] 김진대[2] 남재창[1]

[1]한동대학교 전산전자공학부
[2]서울과학기술대학교 컴퓨터공학과

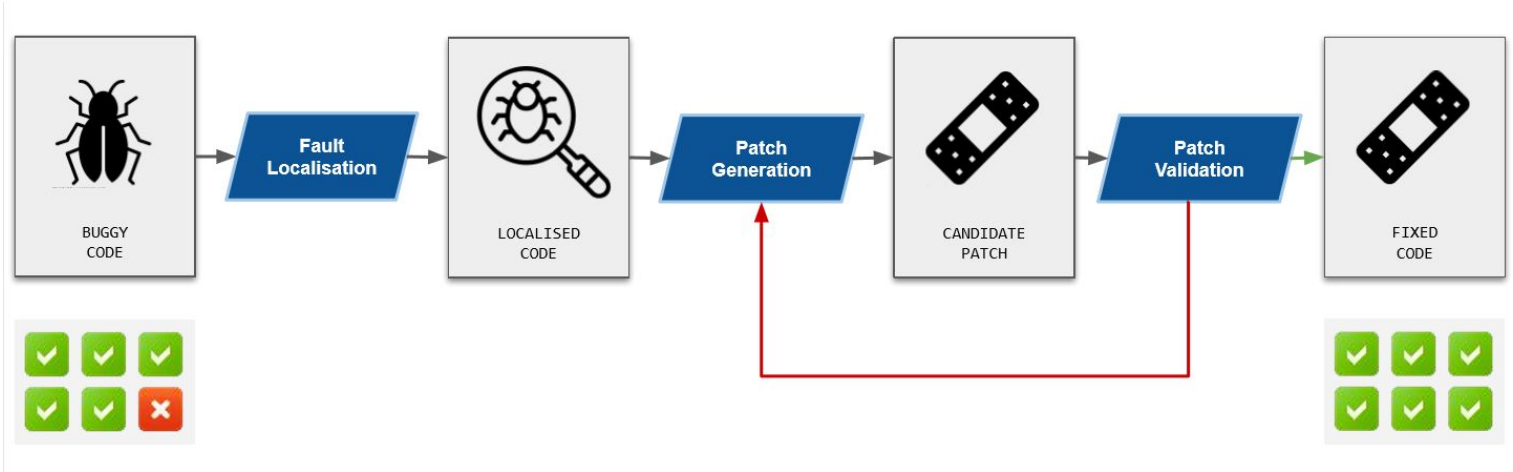
지능형 소프트웨어 공학 연구실 (ISEL)

Contents

- Introduction
- Approach
- Experimental Setup
- Experiment Result
- Conclusion & Future Works

Introduction

Heuristic-based Automated Program Repair [1]



Search Spaces

All ranges considered for generating candidate patches [2]

- Bug fixing operations
- Bug location
- Ingredients
- Test Suite

Search Spaces

Operation

- Insert
- Move
- Delete
- Update

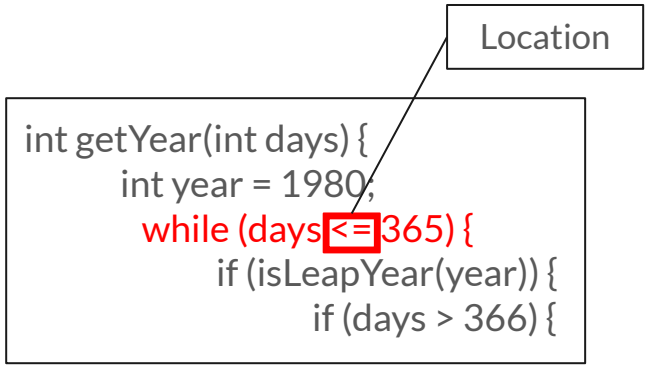
Location

- Code block
- Line
- Variable

Ingredient

- Source code from other parts of the same project
- Source code from external projects

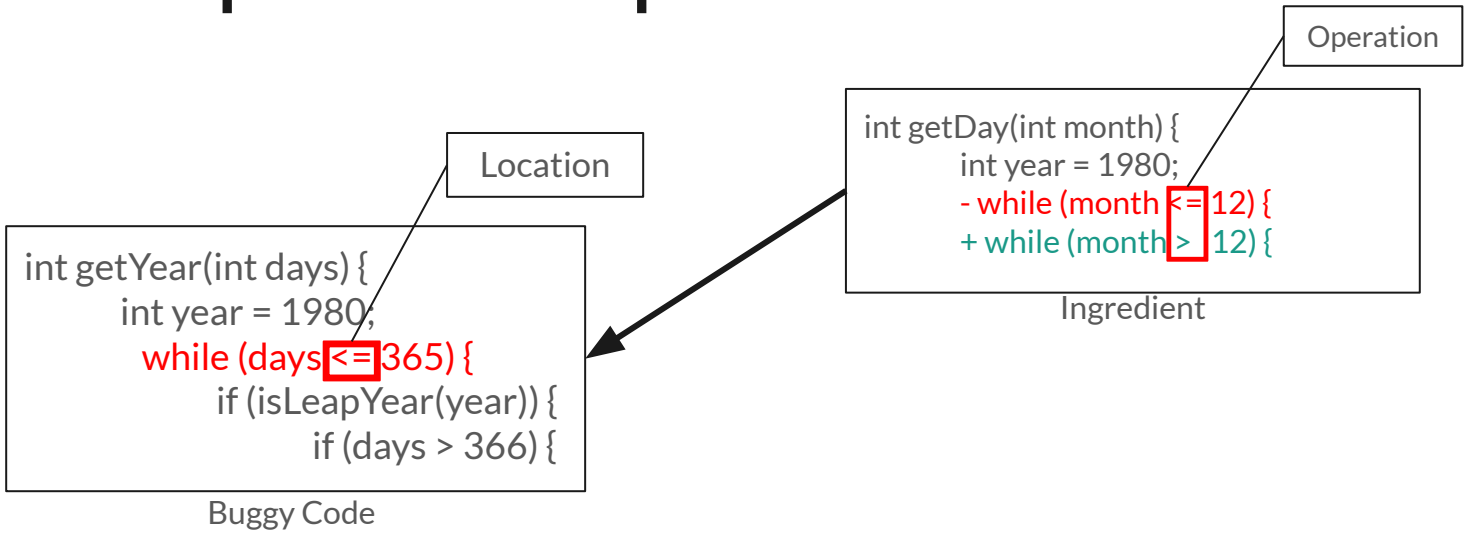
Search Spaces: Example



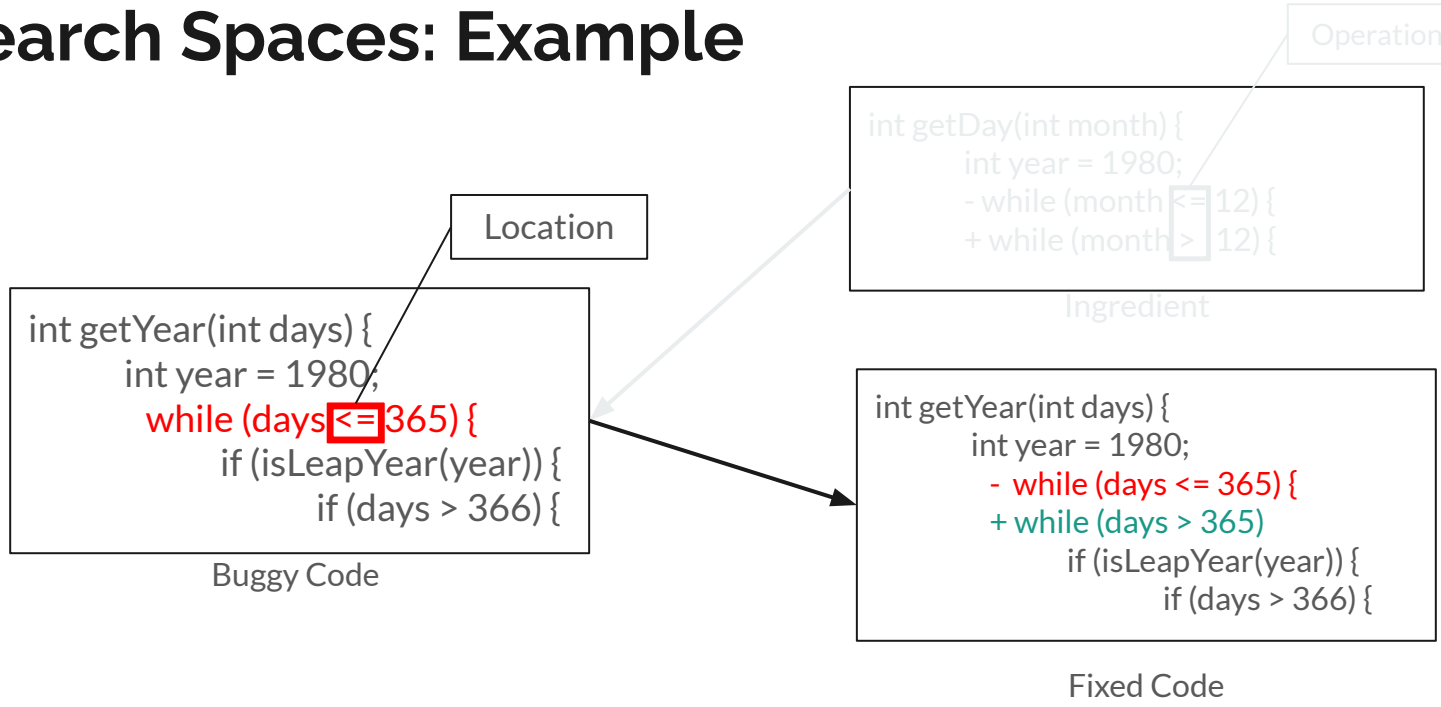
Buggy Code

[1] The Hitchhiker's Guide to Search-Based Program Repair, Chris Timperley, Carnegie Mellon University

Search Spaces: Example



Search Spaces: Example



Search Spaces Problem

The performance of APR tools is directly related to the search space [2]

- Existing APR techniques still have vast patch search spaces
- Expanding the search space leads to resource wastage
- The density of correct patches decreases with more candidate patches

[2] Fan Long and Martin Rinard. 2016. “An Analysis of the Search Spaces for Generate and Validate Patch Generation Systems.” In IEEE/ACM 38th International Conference on Software Engineering (2016). 702–713.

How can we effectively decrease search space?

Search Spaces

Operation

- Insert
- Move
- Delete
- Update

Location

- Code block
- Line
- Variable

Ingredient

- Source code from other parts of the same project
- Source code from external projects

Two bugs with *similar* **bug-introducing changes (BIC)**
will also have *similar* **fixing commits (BFC)**

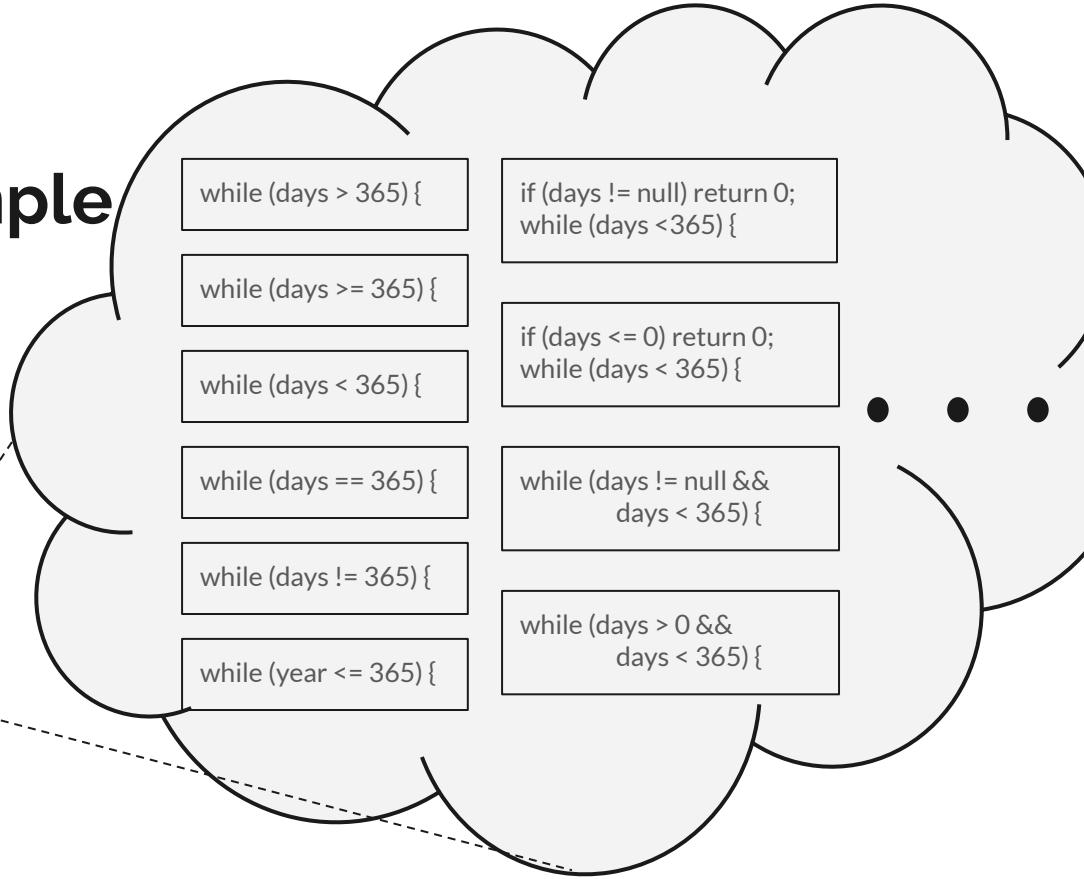
Search Space: Example

Before Buggy Code

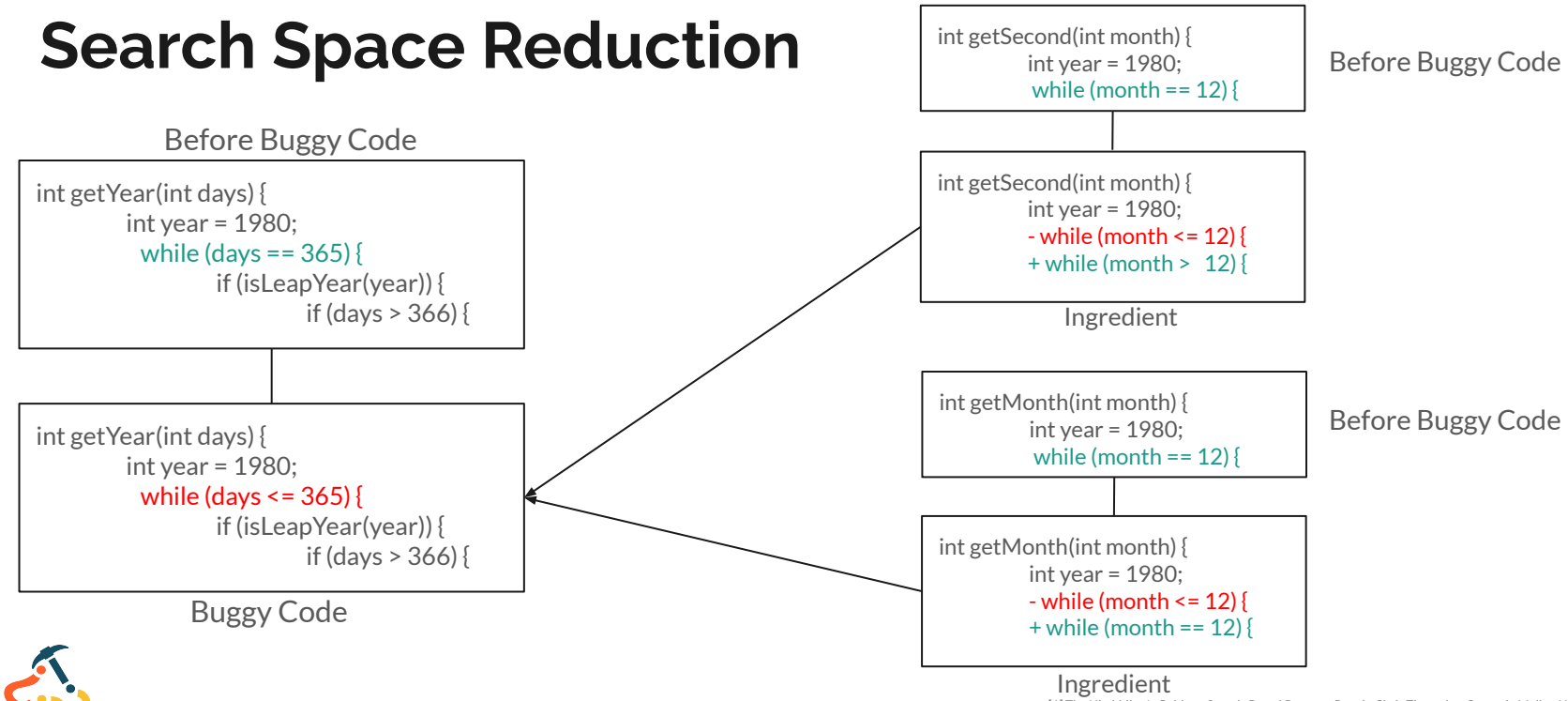
```
int getYear(int days) {
        int year = 1980;
          while (days == 365) {
                  if (isLeapYear(year)) {
                          if (days > 366) {
```

```
int getYear(int days) {
        int year = 1980;
          while (days <= 365) {
                  if (isLeapYear(year)) {
                          if (days > 366) {
```

Buggy Code



Search Space Reduction



Related work

Boosting Automated Program Repair with Bug-Inducing Commits (Ming et al., ICSE-NEIR ‘20) [3]

- Repair bugs by learning from how they were introduced rather than from how other bugs were fixed
- **Fixing ingredients needed to fix a bug can be inferred from the commit that introduced the bug**

[4] Zhang, Xindong, et al. "Precfix: Large-scale patch recommendation by mining defect-patch pairs." Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice. 2020.

Related work

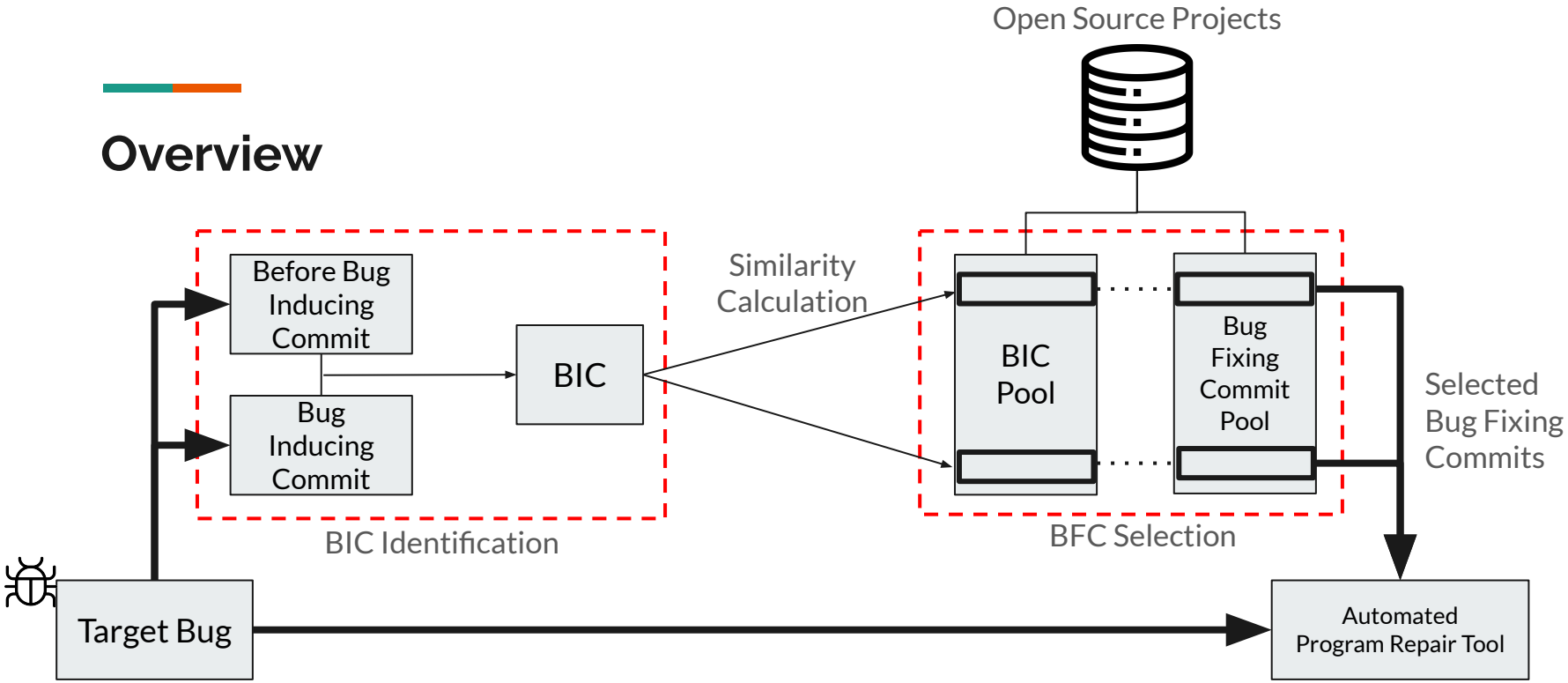
Precfix (Zhang et al., ICSE-SEIP ‘20) [4]

- Collecting defect-patch pairs from development histories
- Perform clustering and extracts generic reusable patching patterns
- **Typical developer behaviors in committing bug fixes**

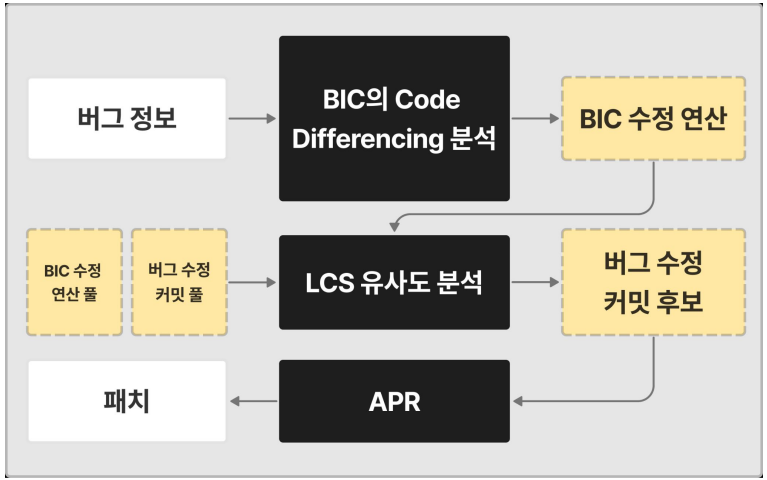
[4] Zhang, Xindong, et al. "Precfix: Large-scale patch recommendation by mining defect-patch pairs." Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice. 2020.

Approach

Overview

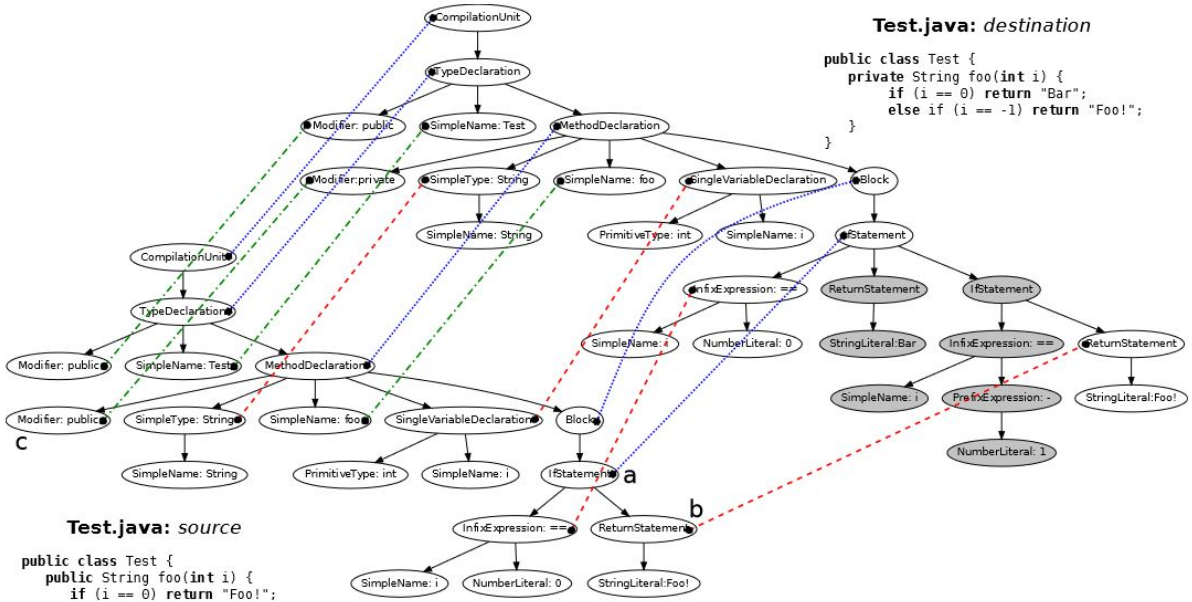

Open Source Projects
Before Bug Inducing Commit
Bug Inducing Commit
BIC
BIC Identification
Similarity Calculation
BIC Pool
Bug Fixing Commit Pool
BFC Selection
Selected Bug Fixing Commits
Target Bug
Automated Program Repair Tool

SPI: Similar Patch Identifier for Automated Program Repair



- **Identify bug introducing change** of provided bug
- **Extract bug fixes operations with similar bug introducing change**
- Use heuristic-based automated program repair tool **with reduced search space**

GumTree: Code Differencing Tool (Falleri et al. ASE 2014)



[6] Jean-Rémy Falleri, Floréal Morandat, Xavier Blanc, Matias Martinez, and Martin Monperrus. "Fine-Grained and Accurate Source Code Differencing". 29th ACM/IEEE International Conference on Automated Software Engineering (2014), 313–324.

GumTree: Closure-14 Example

```
} else if (parent.getLastChild() == node){
  if (cfa != null) {
    for (Node finallyNode : cfa.finallyMap.get(parent)) {
      cfa.createEdge(fromNode, Branch.ON_EX, finallyNode);
    }
  }
  return computeFollowNode(fromNode, parent, cfa);
```

Before Bug Inducing Commit

```
} else if (parent.getLastChild() == node){
  if (cfa != null) {
    for (Node finallyNode : cfa.finallyMap.get(parent)) {
      cfa.createEdge(fromNode, Branch.UNCOND, finallyNode);
    }
  }
  return computeFollowNode(fromNode, parent, cfa);
```

After Bug Inducing Commit

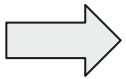
```
[===
insert-node
---
Modifier: final [1356,1361]
to
TypeDeclaration [1282,35017]
at 1, ===
insert-node
---
TYPE_DECLARATION_KIND: class [1362,1367]
to
TypeDeclaration [1282,35017]
at 2, ===
insert-node
---
SimpleName: ControlFlowAnalysis [1368,1387]
to
TypeDeclaration [1282,35017]
at 3, ===
update-node
---
QualifiedName: Branch.ON_EX [26383,26395]
replace Branch.ON_EX by Branch.UNCOND, ===
delete-node
---
Modifier: final [1356,1361]
===, ===
delete-node
---
TYPE_DECLARATION_KIND: class [1362,1367]
===, ===
delete-node
---
SimpleName: ControlFlowAnalysis [1368,1387]
===]
```

<Snippet of Closure-14 Bug Introducing Change>

Numerical Vector of Change Operation

- Convert the **GumTree result** into numerical vectors
- Each node type and edit operation has specific value to be distincted

```
[===
insert-node
---
Modifier: final [1356,1361]
to
TypeDeclaration [1282,35017]
at 1, ===
insert-node
---
TYPE_DECLARATION_KIND: class [1362,1367]
to
TypeDeclaration [1282,35017]
at 2, ===
insert-node
---
SimpleName: ControlFlowAnalysis [1368,1387]
to
TypeDeclaration [1282,35017]
at 3, ===
```

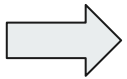


```
public static String[] expanded_nodes = {
                /* 1. */ "ABSTRACT",
                /* 2. */ "ANNOTATION_PROPERTY",
                /* 3. */ "AnnotationTypeDeclaration",
                /* 4. */ "AnnotationTypeMemberDeclaration",
                /* 5. */ "ANNOTATIONS_PROPERTY",
                /* 6. */ "AnonymousClassDeclaration",
                /* 7. */ "ANONYMOUS_CLASS_DECLARATION_PROPERTY",
                /* 8. */ "ARGUMENTS_PROPERTY",
                /* 9. */ "ArrayAccess",
```

...

```
public static int getNodeNum(String str) {
    // on node types
    if (str.equals("delete-node")) {
        return 0;
    }
    if (str.equals("insert-node")) {
        return 1;
    }
    if (str.equals("update-node")) {
        return 2;
    }
```

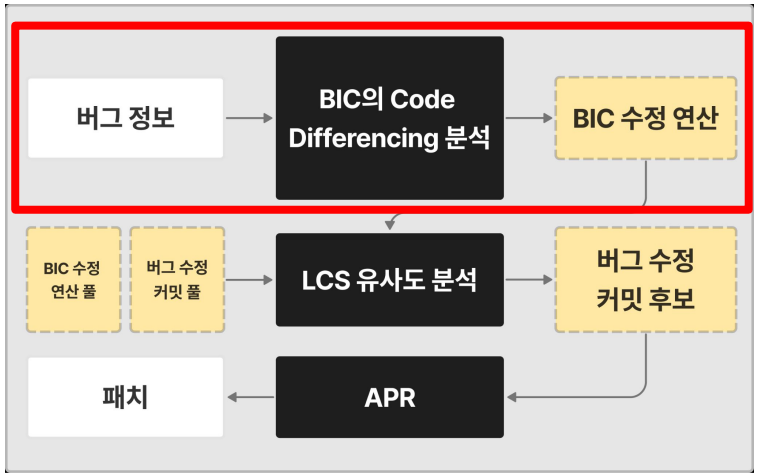
...



```
243,316,338,
```

<Snippet of Closure-14 Bug Introducing Change>

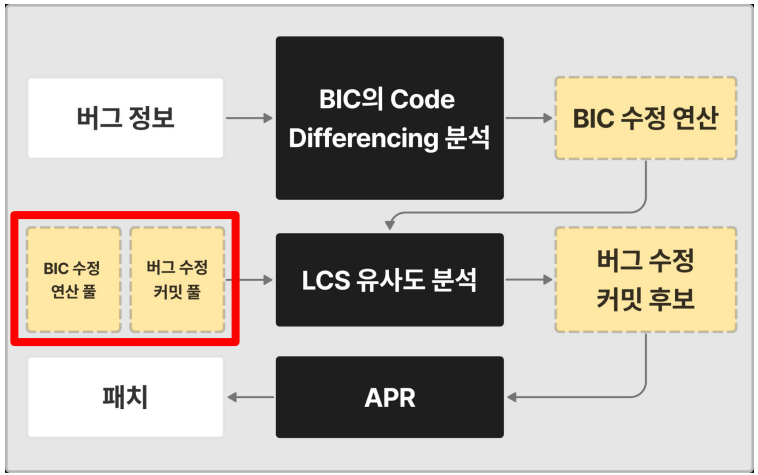
Identify Bug Introducing Change of Target Bug



- Use git blame command
- Define into change operation[5] to seek similar BIC
- Use GumTree[6] to extract edit operation in Abstract Syntax Tree
- Convert to numerical vector

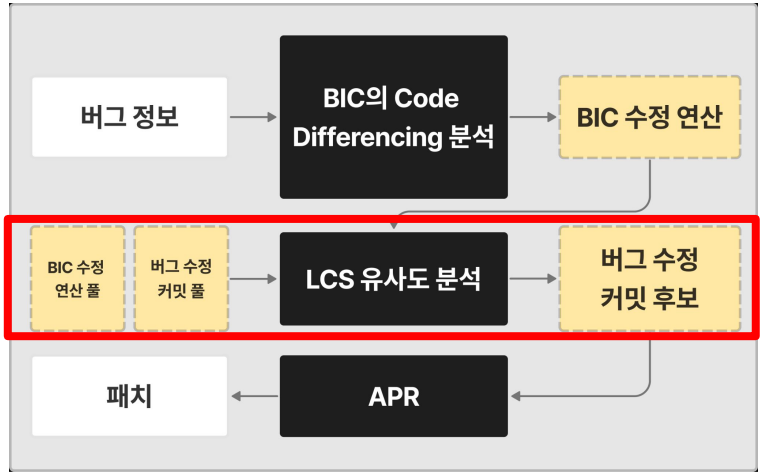
[5] Connor, Aidan, Aaron Harris, Nathan Cooper, and Denys Poshyvanyk. "Can we automatically fix bugs by learning edit operations?." IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), (2012) 782-792
[6] Jean-Rémy Falleri, Floréal Morandat, Xavier Blanc, Matias Martinez, and Martin Monperrus. "Fine-Grained and Accurate Source Code Differencing". 29th ACM/IEEE International Conference on Automated Software Engineering (2014), 313–324.

Bug Introducing Change Pool & Bug Fixing Commit



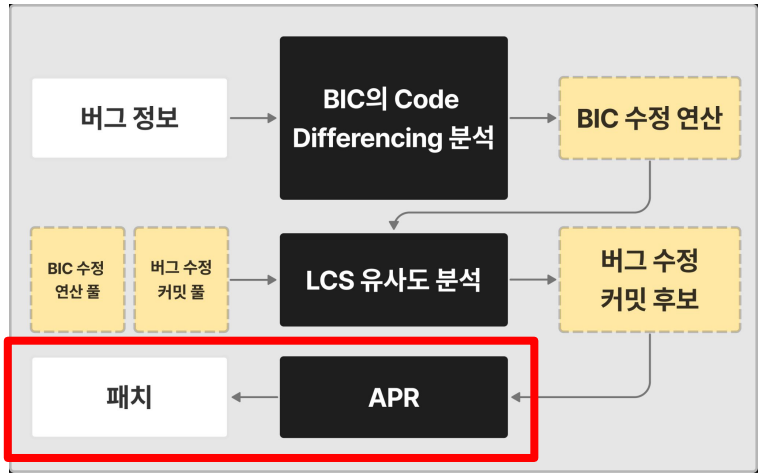
- Collect BIC and Bug Fixing Commits From Open-Source Projects
- Make csv file of BIC numerical Vector and Bug Fixing Commit as pair

Identify Bug Fix with Similar Bug Introducing Change



- Calculate similarity between target bug's BIC and collected BIC using **Longest Common Subsequence (LCS) Algorithm**
- **Extract Top 10 Bug Fixing Commits**

Automated Program Repair with Reduced search space



Provide selected bug fixing commit into APR Tool

Experimental Setup

Research Questions

RQ1. What is the performance of SPI compared to Baseline APR?

RQ2. Does pool size affect the performance?

RQ3. Does LCS similarity for patch effectively provide the necessary modifications?

Experimental Setup

Baseline APR: ConFix[7]

Java Bug Benchmark: Defects4J[8]

- Chart, Closure, Lang, Math, Time, Mockito

[7] Kim, Jindae, and Sunghun Kim. “Automatic patch generation with context-based change application.” Empirical Software Engineering 24.6 (2019): 4071-4106.
[8] Just, R, Jalali, D., Ernst, M.D. “Defects4J: A database of existing faults to enable controlled testing studies for Java programs.” Proceedings of the 2014 International Symposium on Software Testing and Analysis. 2014.

Experimental Setup

Fault Localization: Perfect Fault Localization

Pool: Apache open source project

- Beam, Cassandra, Hadoop, jUDDI, Kafka, Spark

[7] Kim, Jindae, and Sunghun Kim. “Automatic patch generation with context-based change application.” Empirical Software Engineering 24.6 (2019): 4071-4106.
[8] Just, R, Jalali, D., Ernst, M.D. “Defects4J: A database of existing faults to enable controlled testing studies for Java programs.” Proceedings of the 2014 International Symposium on Software Testing and Analysis. 2014.

ConFix: Automatic Patch Generation with Context-based Change Application

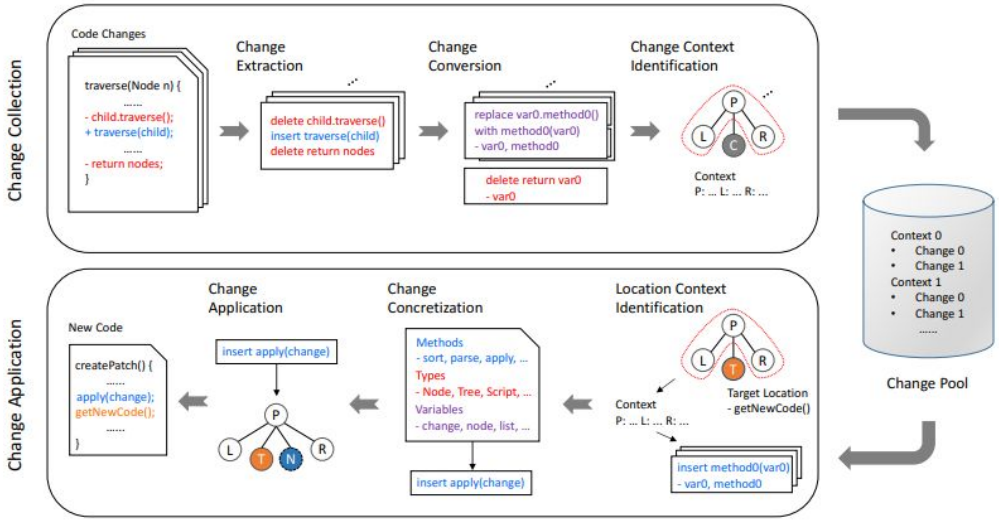


Fig. 1: The Overview of Context-based Change Application Technique

[7] Kim, Jindae, and Sunghun Kim. "Automatic patch generation with context-based change application." Empirical Software Engineering 24.6 (2019): 4071-4106.

Experiment Result

RQ1. What is the performance of SPI compared to Baseline APR?

표 1. 성공적으로 생성한 패치 결과

ConFix With Answer는 ConFix에 정답 수정 커밋이 제공되었을 때를 의미.

Project	SPI	Original ConFix	ConFix with Answers
Chart	**3(4)**	4	N/A
Closure	**3(12)**	6	20
Lang	**2(4)**	5	5
Math	**4(11)**	6	20
Time	**0(1)**	1	1
Mockito	**0(0)**	N/A	4
Sum	**12(32)**	**22**	**50**

- Original ConFix[7]
- ConFix with Answers[8]
- SPI with ConFix generated **12 correct patches**

[7] Kim, Jindae, and Sunghun Kim. "Automatic patch generation with context-based change application." Empirical Software Engineering 24.6 (2019): 4071-4106.
[8] Sunghyun Choi, Junghyun Heo, Chaewoo Yu, and Jaechang Nam. "Analysis on Results of ConFix Execution Through Correct Patch Change Information" KCC 2022.

RQ1. What is the performance of SPI compared to Baseline APR?

0.15%

$$\frac{\text{SPI bug fix commit candidates count}}{\text{ConFix bug fix commit candidates count}} = \frac{10}{6{,}485}$$

RQ1. What is the performance of SPI compared to Baseline APR?

4 additional correct patches

Closure 10
Closure 86
Lang 59
Math 59

RQ2. Does pool size affect the performance?

표 2. 파일 수정 개수가 커질때 SPI가 생성하는 패치의 개수

Project	File Change: 3,262	File Change: 26,660
Chart	3(4)	**3(4)**
Closure	3(8)	**3(12)**
Lang	2(3)	**2(4)**
Math	3(7)	**4(11)**
Time	0(1)	**0(1)**
Mockito	0(0)	**0(0)**
Sum	**11(23)**	**12(32)**

Increasing the number of file modifications considered by SPI leads to **more bugs being fixed**

RQ3. Does LCS similarity for patch effectively provide the necessary modifications?

표 3. LCS 평균 점수의 차이
*: 통계적으로 유의미한 차이를 보임

Project	Patch-Generated LCS Score	No Patch-Generated LCS Score	Overall LCS Score
Chart	0.88	0.75	0.80
Closure	0.83	0.63	0.68
Lang	0.83	0.51	0.60
Math	0.81	0.70	0.73
Time	0.96	0.50	0.75
Mockito	N/A	0.75	0.65
All	0.82*	0.65*	0.70

- **LCS similarity effectively provided the necessary fix operations**
- **Mann-Whitney U Test:** Calculated as 0.00018, indicating high statistical significance at the 5% level

Conclusion & Future Works

Conclusion

Utilized Bug Introducing Change similarity to reduce Search Space

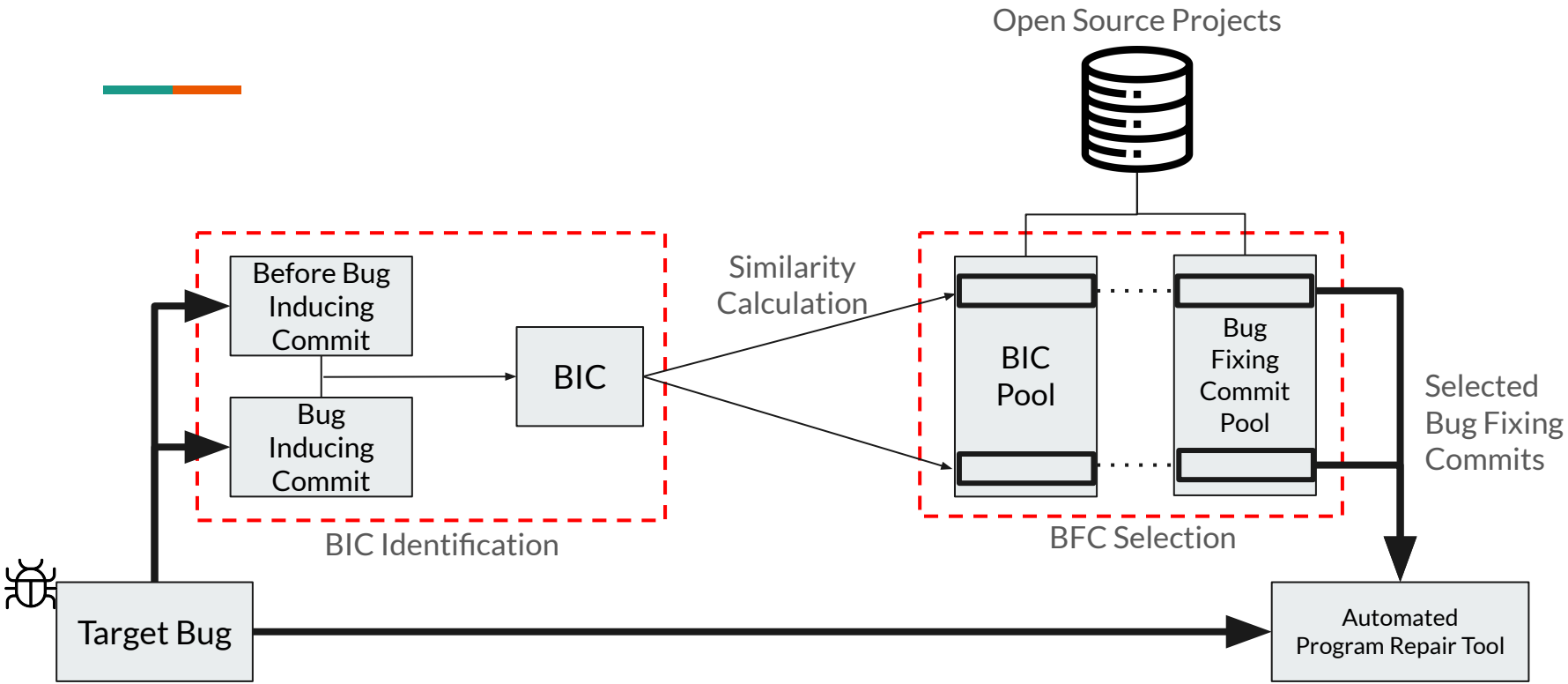
Confirmed reduction in search space and performance improvement

Future Works

Build a pool with larger GitHub open source projects with better quality

Try to integrate SPI with other APR

Q & A


Open Source Projects
Before Bug Inducing Commit
Bug Inducing Commit
BIC
BIC Identification
Similarity Calculation
BIC Pool
Bug Fixing Commit Pool
BFC Selection
Selected Bug Fixing Commits
Target Bug
Automated Program Repair Tool

Bug Fixing Commit / Bug Introducing Change Pool

선택된 Apache 오픈소스 프로젝트 내 버그 수정 Commit

```
a960d87e05f01000a758d8e7e5a58be57d13eb33,3edacd632c2f2c360ca1d931c6b4f6fd7326511f,src/java/org/apache/cassandra/service/MigrationCoordinator.java,sr
```

...

위 Commit과 대응되는 BIC 수정 연산의 수치형 벡터

```
316,272,316,437,442,750,846,750,846,750,846,782,782,782,782,847,442,168,102,656,583,583,678,612,661,612,661,612,597,613,612,597,613,612,597,613,612,613
```

...