FAngelix, Verifix, and Poracle

2022년 2월

소프트웨어재난연구센터 겨울 워크샵

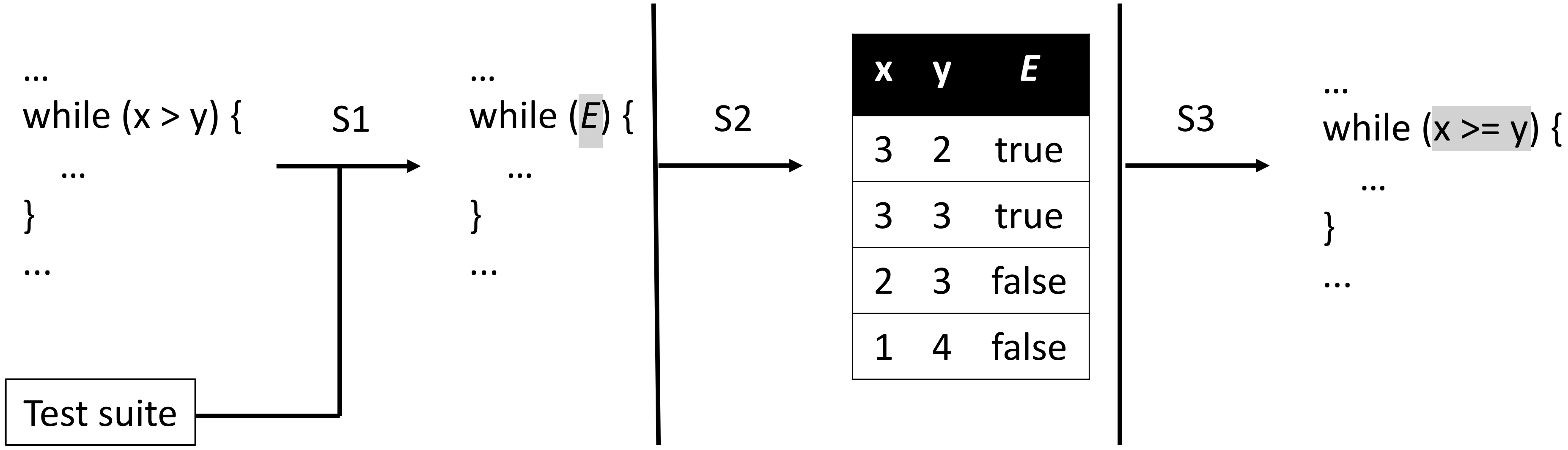
이주용 (UNIST)

Speeding up Constraint-Based Program Repair Using a Search-Based Technique

Jooyong Yi and Elkhan Ismayilzada

Information and Software Technology, 2022

Constraint-Based Program Repair (Angelix)



```
…
while (x > y) {
    …
}
…
```

```
…
while (E) {
    …
}
…
```

x	y	*E*
3	2	true
3	3	true
2	3	false
1	4	false

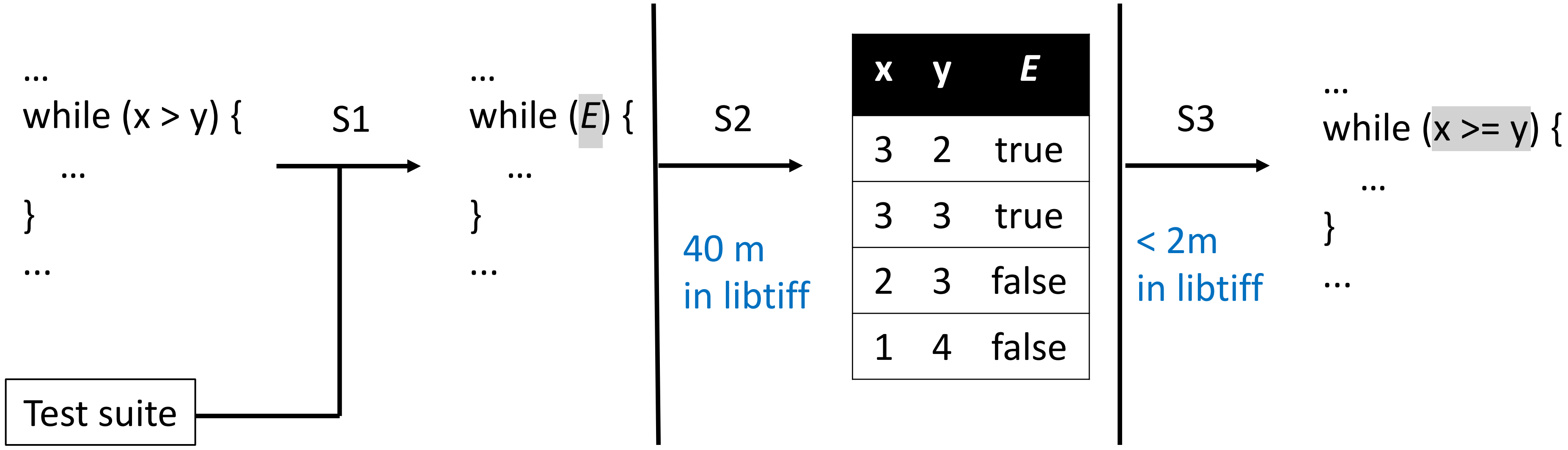
```
…
while (x >= y) {
    …
}
…
```

S1. A search for suspicious expressions (via statistical fault localization)

S2. A search for the specification of the identified suspicious expressions

S3. A search for patch expressions that satisfies the extracted specification

Angelix is unnecessarily slow



```
...
while (x > y) {
    ...
}
...
```

Test suite

S1

```
...
while (E) {
    ...
}
...
```

S2

40 m
in libtiff

x	y	*E*
3	2	true
3	3	true
2	3	false
1	4	false

S3

< 2m
in libtiff

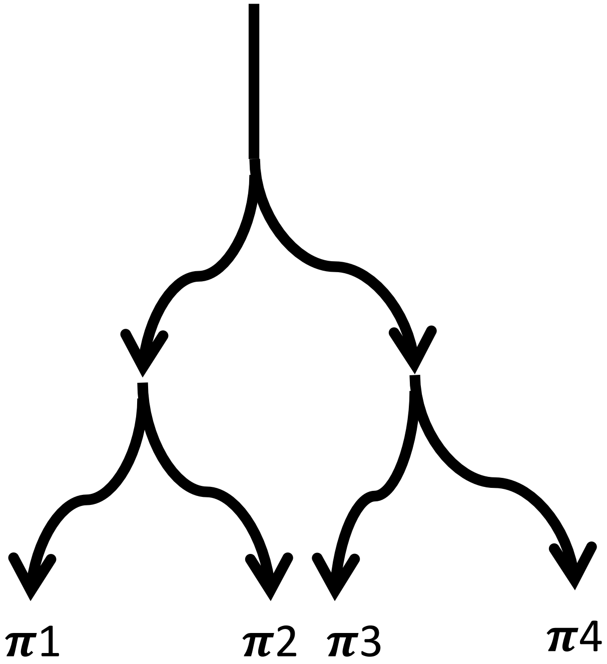
```
...
while (x >= y) {
    ...
}
...
```

Why Slow?

```
…
if (x < y)
  x = y;
if (x < y)
  z = foo(x);
…
```

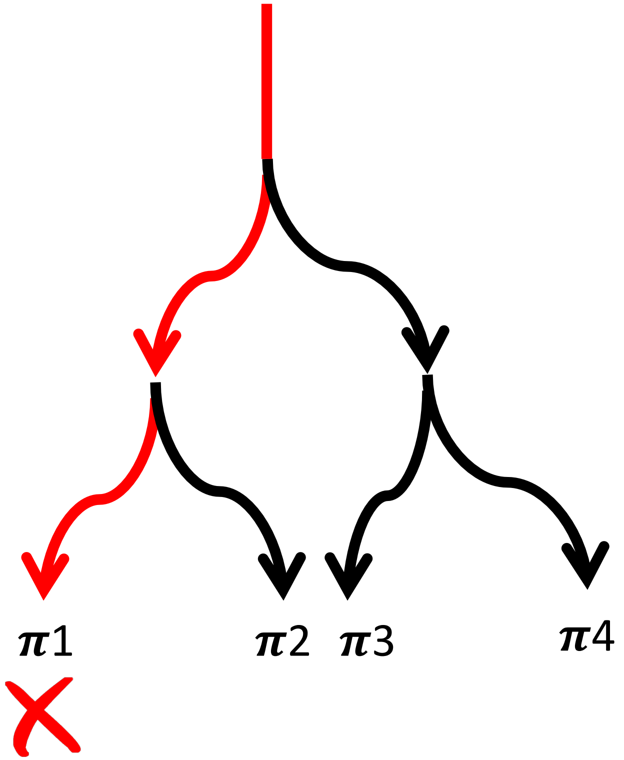
Why Slow?

```
…
if (α)
  x = β;
if (γ)
  z = foo(x);
…
```



Why Slow?

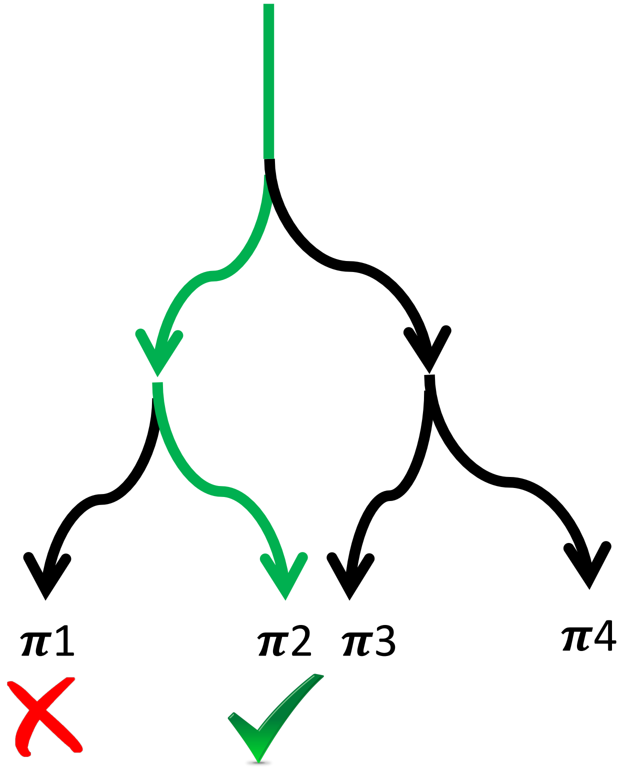
```
…
if (α)
   x = β;
if (γ)
   z = foo(x);
…
```



Why Slow?

```
…
if (α)
   x = β;
if (γ)
   z = foo(x);
…
```

path	α	β	γ
π2	T	0	F

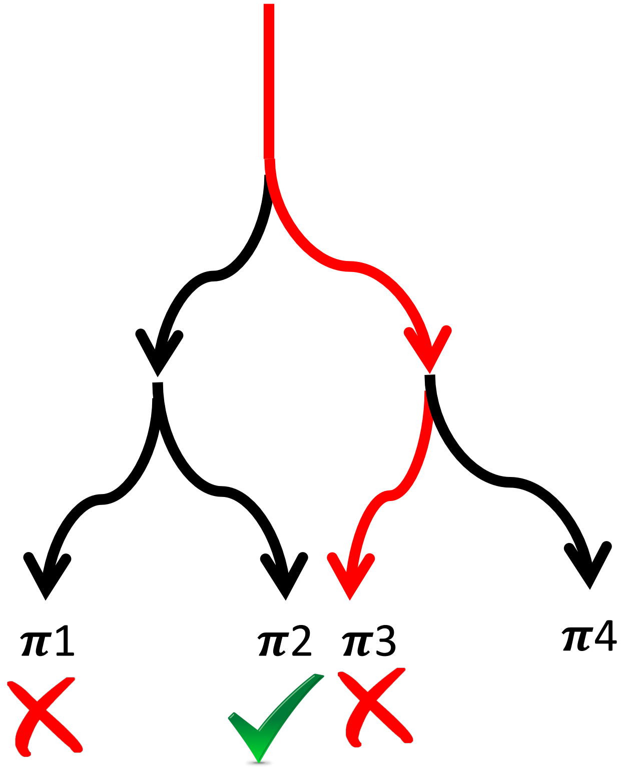


angelic path

Why Slow?

```
…
if (α)
   x = β;
if (γ)
   z = foo(x);
…
```

path	α	β	γ
$\pi2$	T	0	F

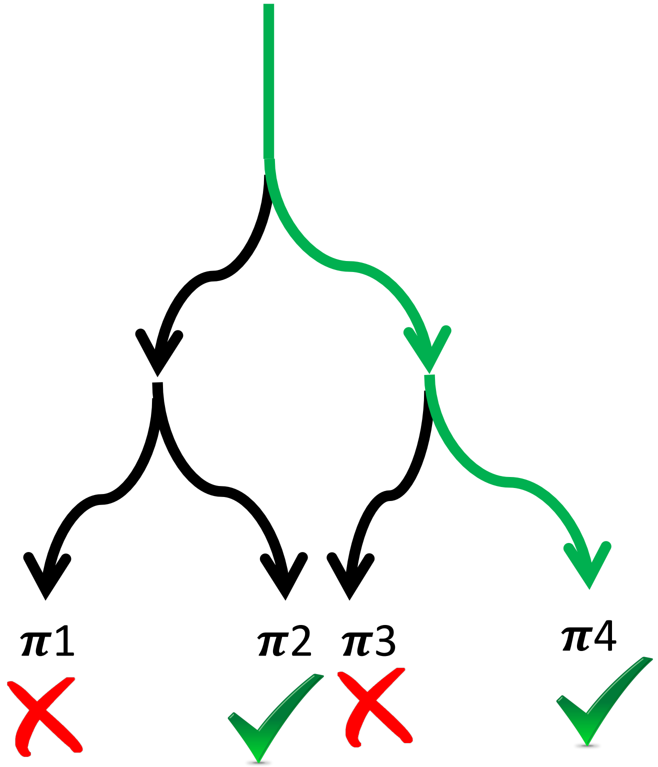


Why Slow?
Reason 1: Angelix performs exhaustive search

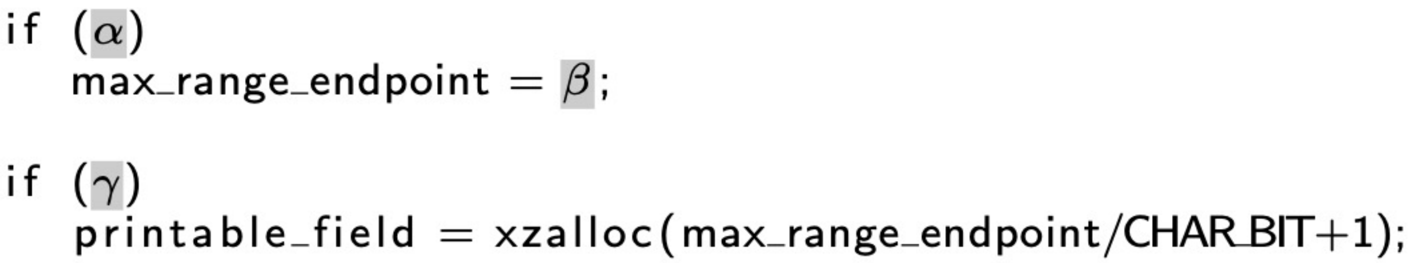
```
…
if (α)
   x = β;
if (γ)
   z = foo(x);
…
```

path	α	β	γ
$\pi 2$	T	0	F
$\pi 4$	F	-	F

angelic forest

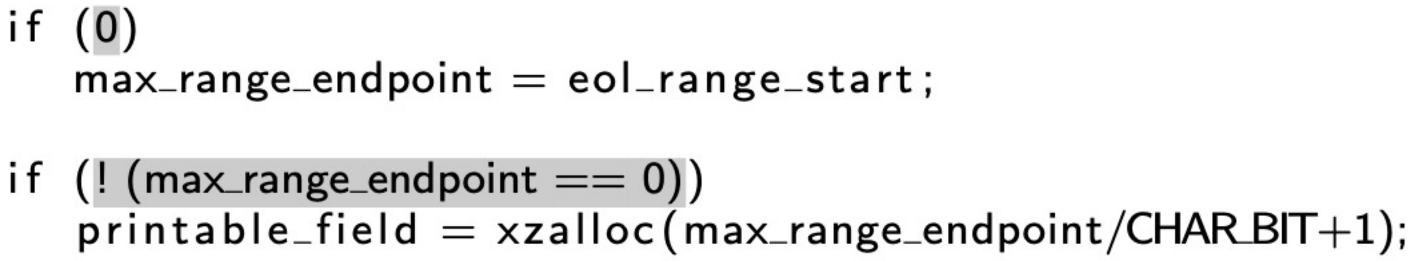


Angelix performs exhaustive search to find a minimal repair



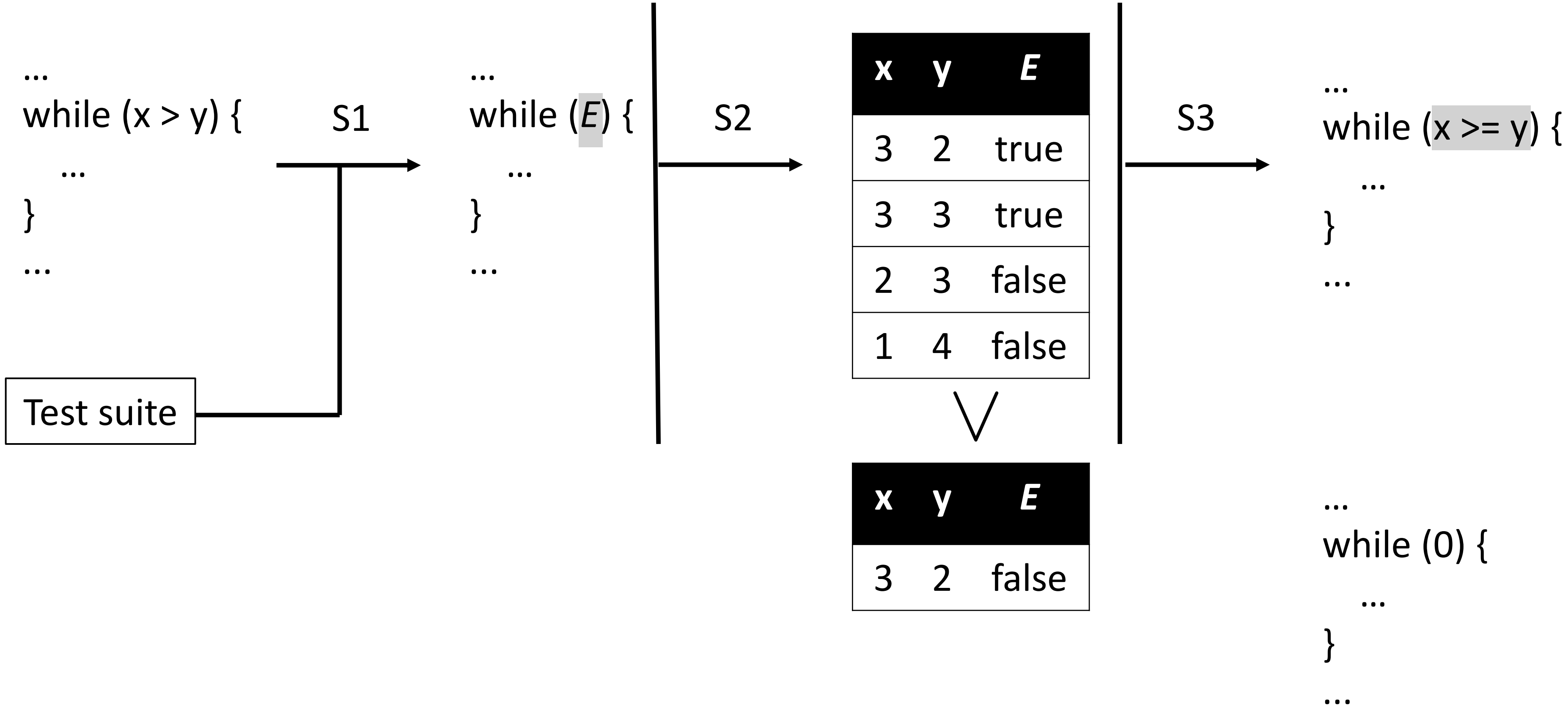
```
if (α)
   max_range_endpoint = β;

if (γ)
   printable_field = xzalloc(max_range_endpoint/CHAR_BIT+1);
```

$$\{\pi_1 : \langle(\alpha, False, \sigma_1), (\gamma, False, \sigma_2)\rangle,$$
$$\pi_2 : \langle(\alpha, True, \sigma_3), (\beta, 0, \sigma_4), (\gamma, False, \sigma_5)\rangle\}$$



```
if (0)
   max_range_endpoint = eol_range_start;

if (! (max_range_endpoint == 0))
   printable_field = xzalloc(max_range_endpoint/CHAR_BIT+1);
```

Angelix performs exhaustive search to find a minimal repair



```
…
while (x > y) {
    …
}
…
```

Test suite

S1

```
…
while (E) {
    …
}
…
```

S2

x	y	***E***
3	2	true
3	3	true
2	3	false
1	4	false

$\bigvee$

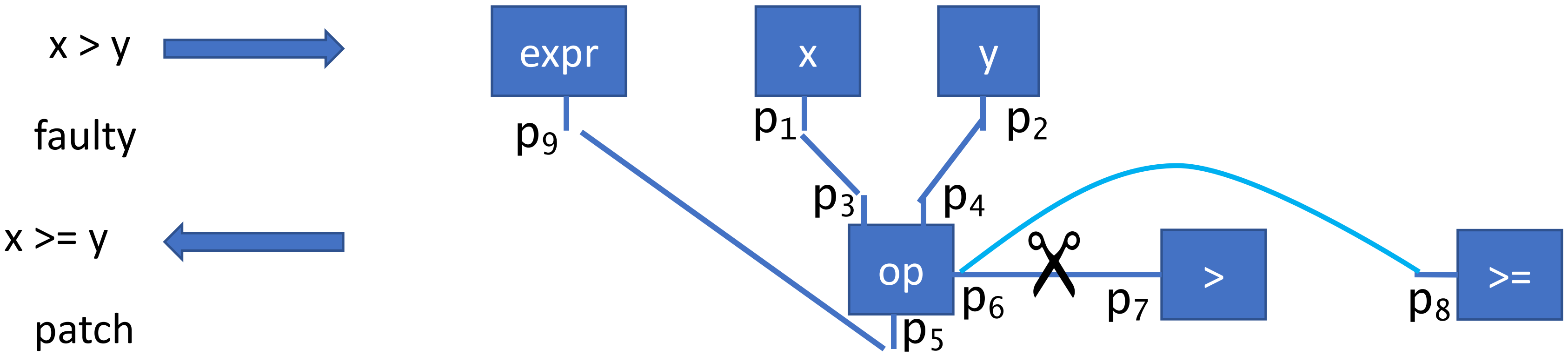
x	y	***E***
3	2	false

S3

```
…
while (x >= y) {
    …
}
…
```

```
…
while (0) {
    …
}
…
```

Finding Minimal Repair via Partial MaxSMT



x > y

faulty

x >= y

patch

$p_6{==}p_8 \models$

$val(p_1){==}x \land val(p_2){==}y \land val(p_9){==}expr$

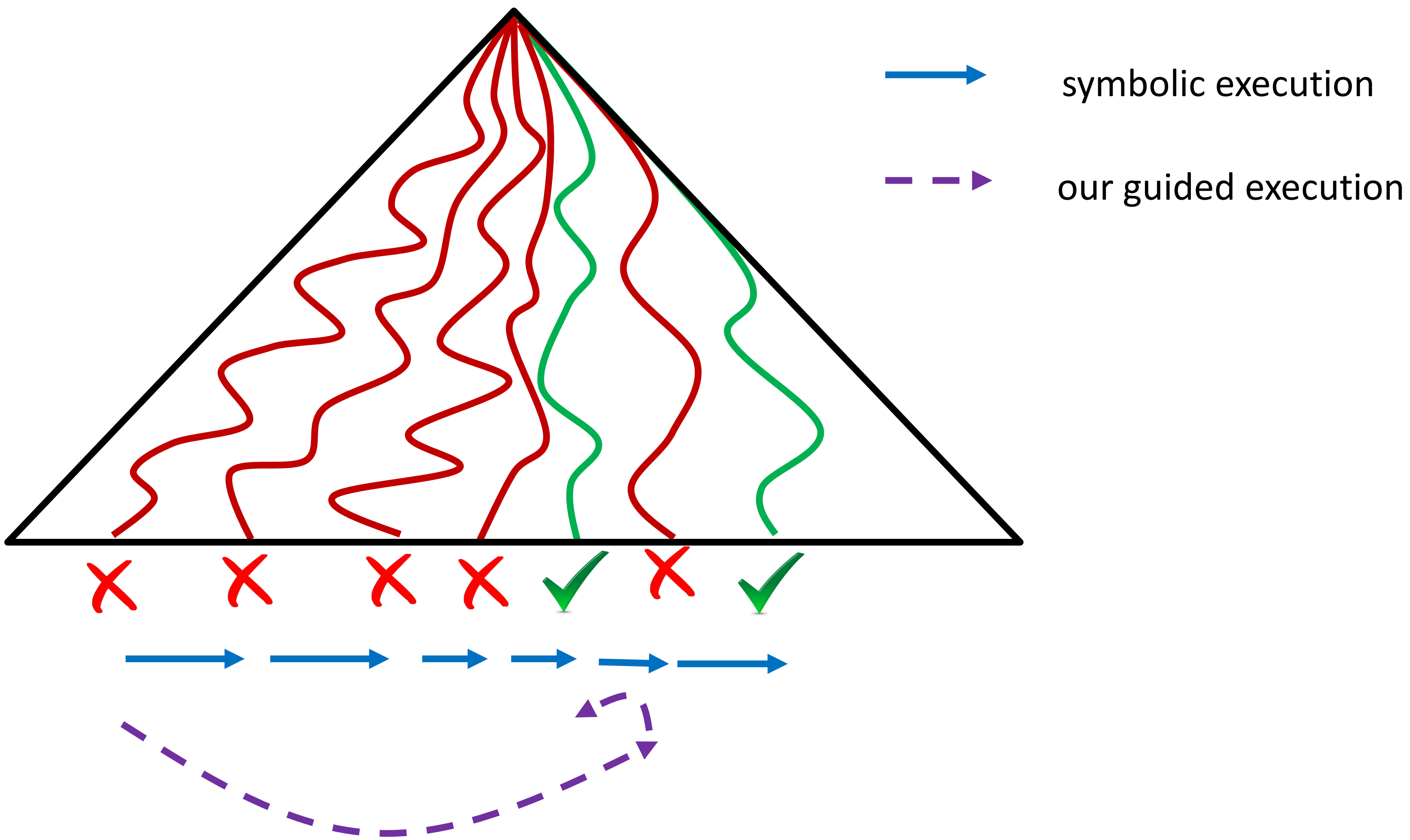
$\land\ (p_6{==}p_7) \implies (val(p_3) > val(p_4)) == val(p_5)$

$\land\ (p_6{==}p_8) \implies (val(p_3) >= val(p_4)) == val(p_5)$

Soft Clauses $\land\ (p_1{==}p_3) \land (p_2{==}p_4) \land (p_6{==}p_7) \land (p_5{==}p_9)$

$\land$ (* inferred spec *)

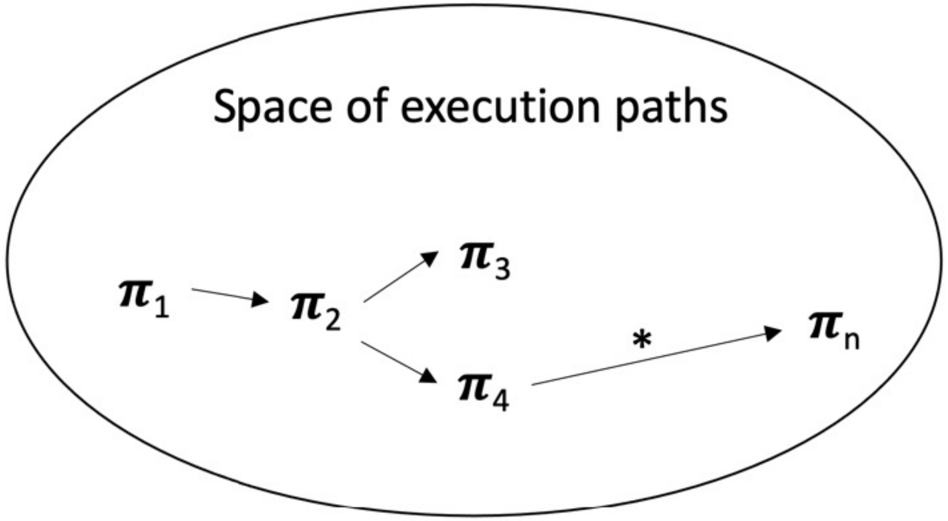
FAngelix idea 1: Guided search



symbolic execution

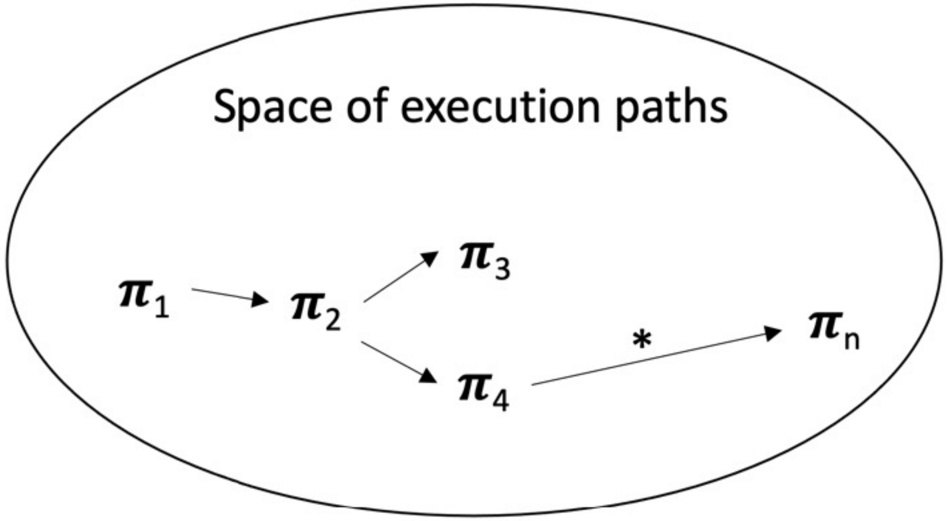
our guided execution

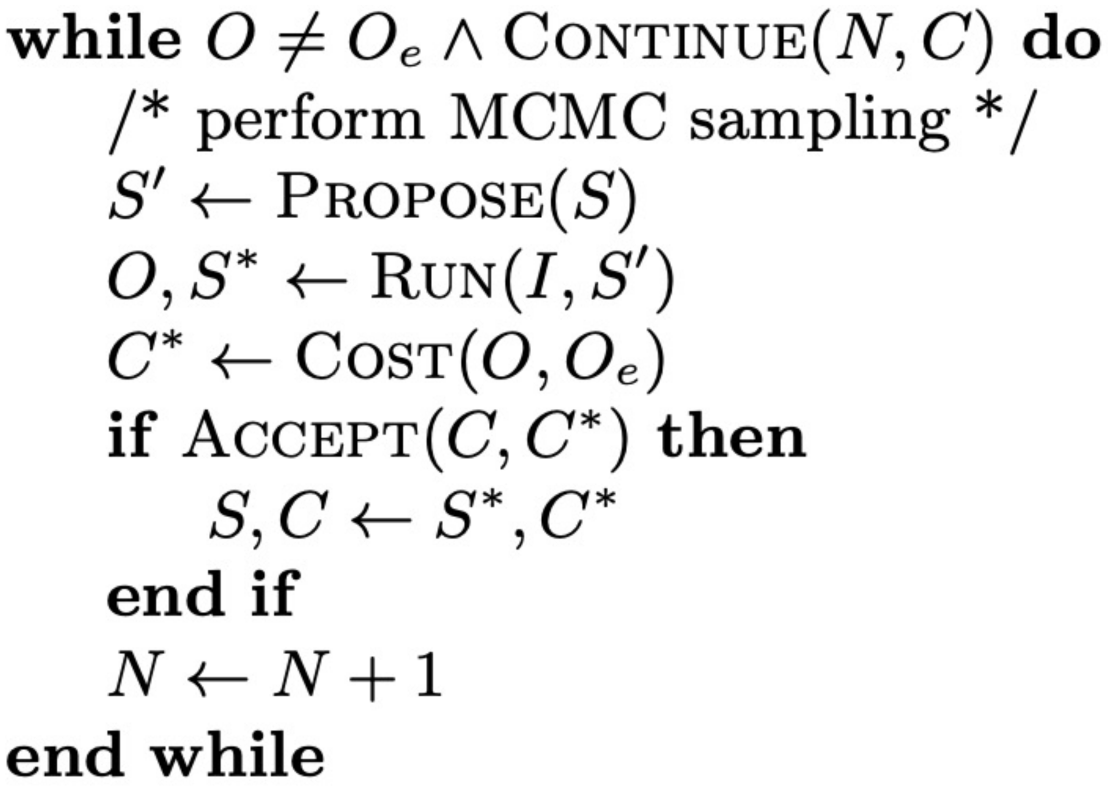
Guided search via MCMC sampling



- π_i 예: {"18-15-18-19": [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]}
- cost가 작아지는 방향으로 유도 (cost가 0이면 angelic path)

Spec Inference Algorithm





```
while $O \neq O_e \wedge \textsc{Continue}(N, C)$ do
    /* perform MCMC sampling */
    $S' \leftarrow \textsc{Propose}(S)$
    $O, S^* \leftarrow \textsc{Run}(I, S')$
    $C^* \leftarrow \textsc{Cost}(O, O_e)$
    if $\textsc{Accept}(C, C^*)$ then
        $S, C \leftarrow S^*, C^*$
    end if
    $N \leftarrow N + 1$
end while
```

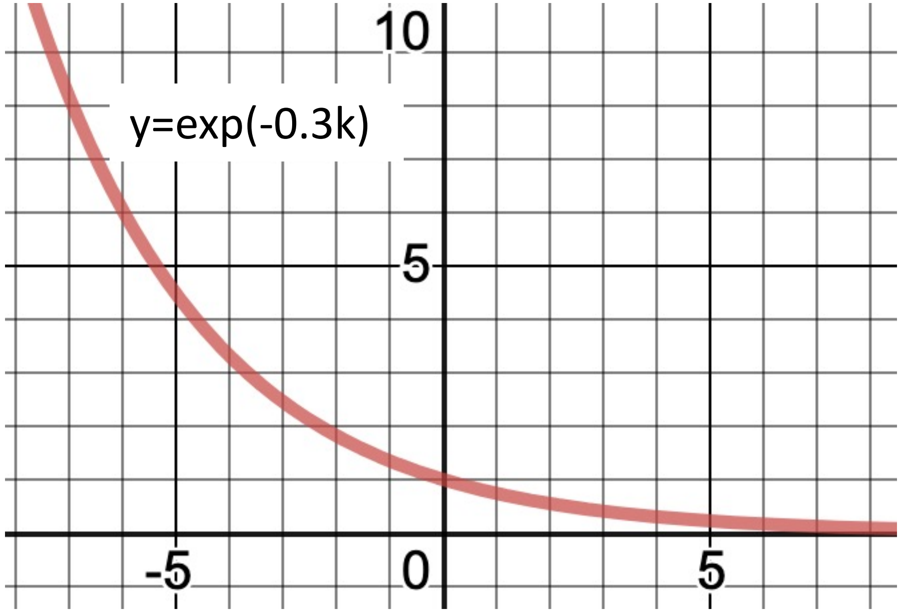
Accept Function

- Metropolis-Hastings acceptance probability with a cost function

$$\alpha(S \to S^*) = \min\left(1, \exp(-\beta \cdot k) \cdot \frac{q(S|S^*)}{q(S^*|S)}\right),$$

where

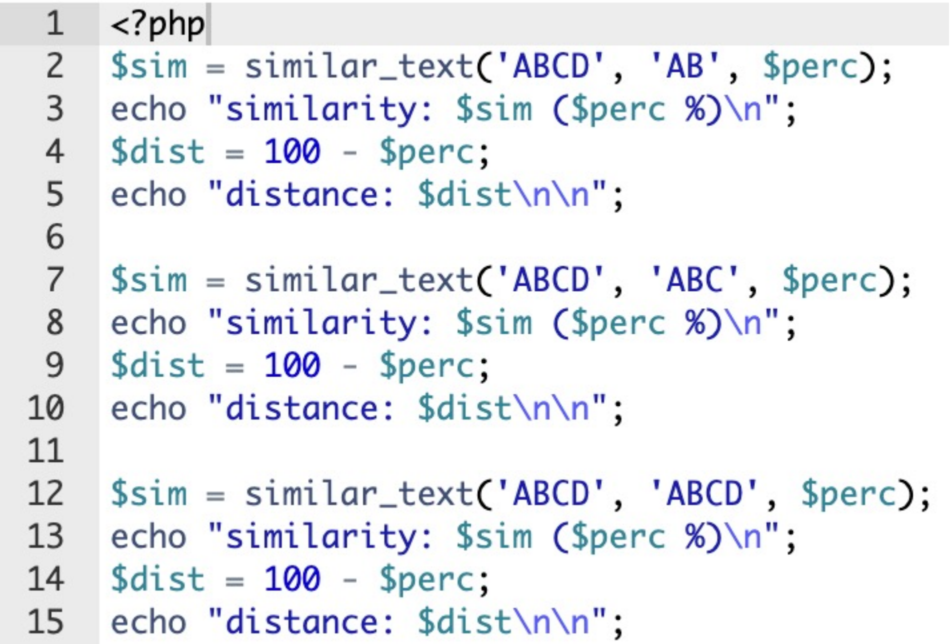
$$k = c(S^*) - c(S).$$



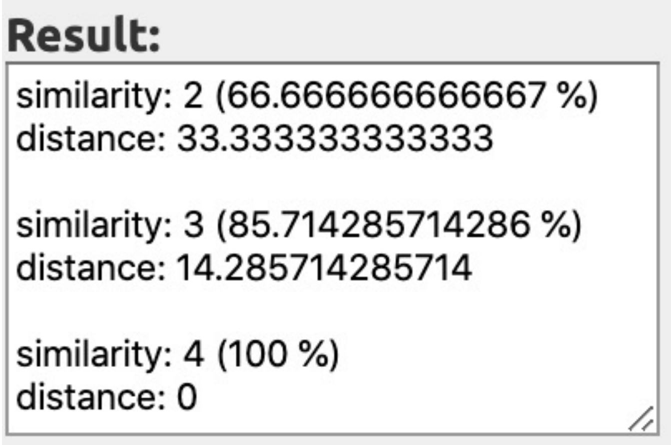
Assignment bugs

- $x = E$ ➔ $x = \alpha$
- Run symbolic execution
- If α does not flow into a conditional expression, solve $Oa(\alpha) = Oe$
- Otherwise,
  - record an executed path as a bit-vector
  - perform a guided random search as before and solve $pc(\alpha) \land Oa(\alpha) = Oe$

An Example of Cost



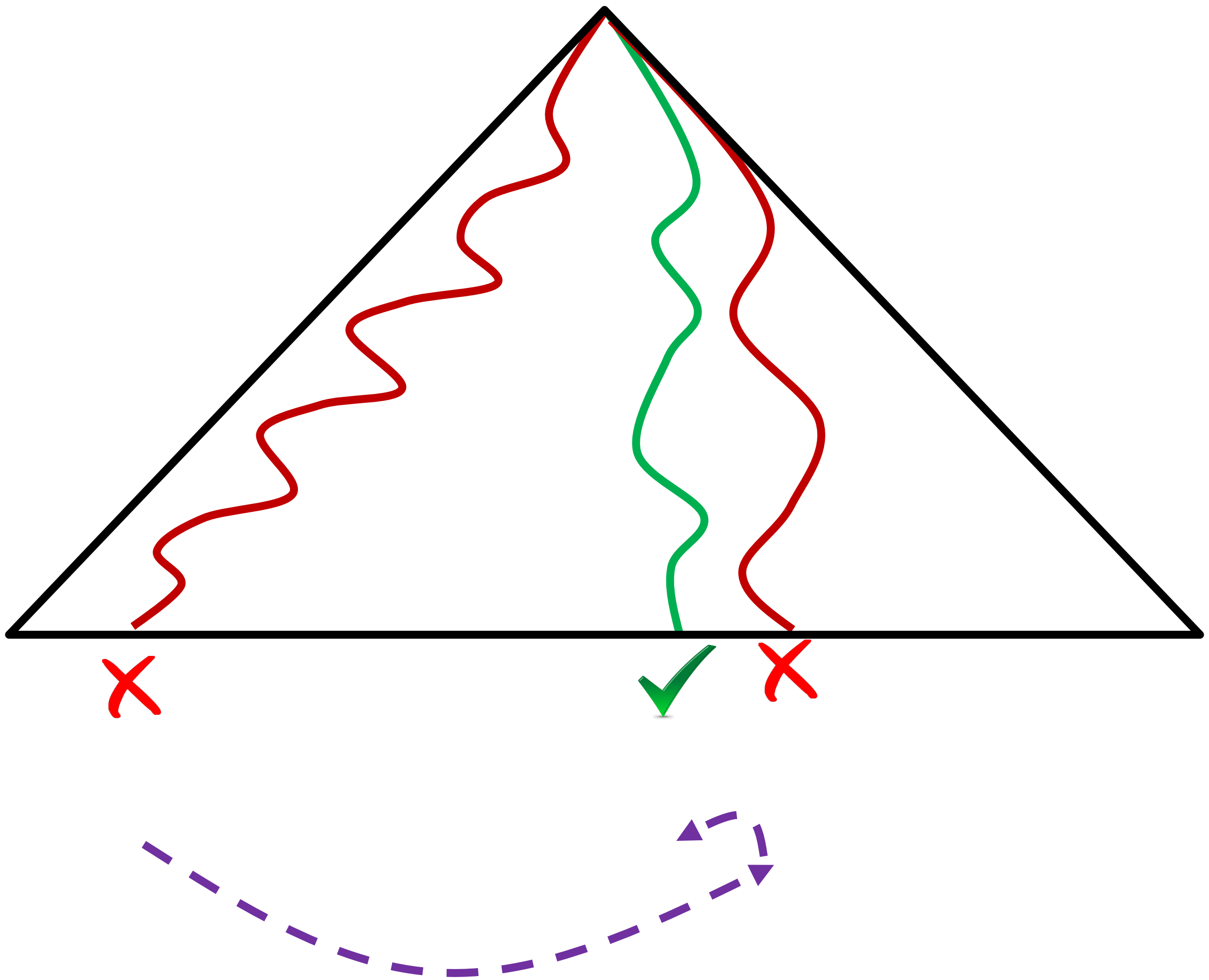
```php
<?php
$sim = similar_text('ABCD', 'AB', $perc);
echo "similarity: $sim ($perc %)\n";
$dist = 100 - $perc;
echo "distance: $dist\n\n";

$sim = similar_text('ABCD', 'ABC', $perc);
echo "similarity: $sim ($perc %)\n";
$dist = 100 - $perc;
echo "distance: $dist\n\n";

$sim = similar_text('ABCD', 'ABCD', $perc);
echo "similarity: $sim ($perc %)\n";
$dist = 100 - $perc;
echo "distance: $dist\n\n";
```



Result:

```
similarity: 2 (66.666666666667 %)
distance: 33.333333333333

similarity: 3 (85.714285714286 %)
distance: 14.285714285714

similarity: 4 (100 %)
distance: 0
```

FAngelix idea 2: No exhaustive search



Angelic path 정제 (refinement)

- 버기 path:

{"18-15-18-19": [1, 1, 0, 0]}

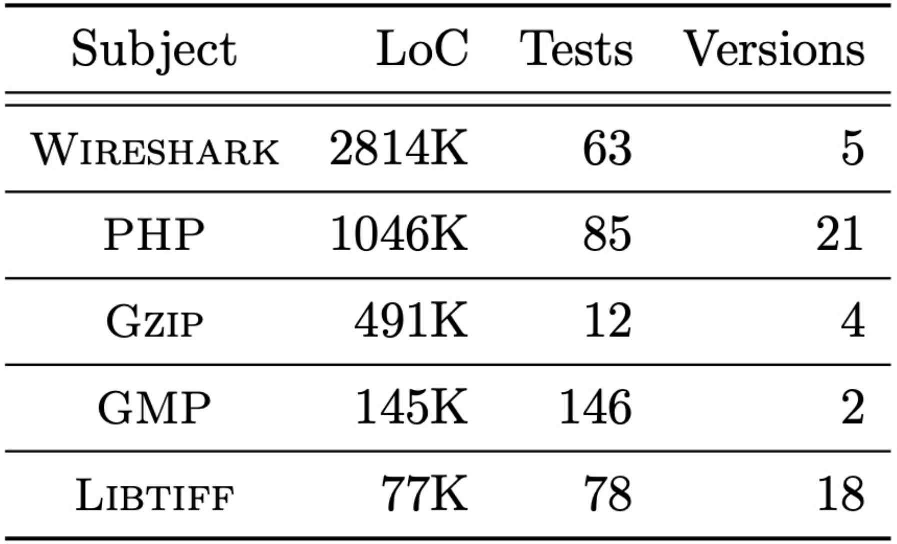
- 찾아진 angelic path:

{"18-15-18-19": [1, **0**, **1**, 0]}

- 정제된 angelic path:

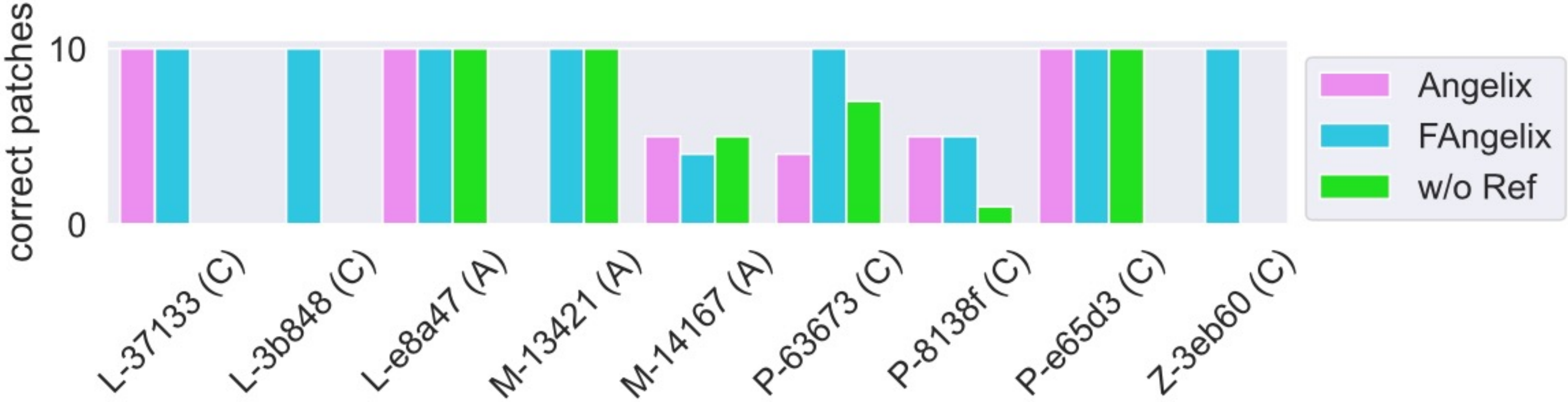
{"18-15-18-19": [1, 1, **1**, 0]}

실험

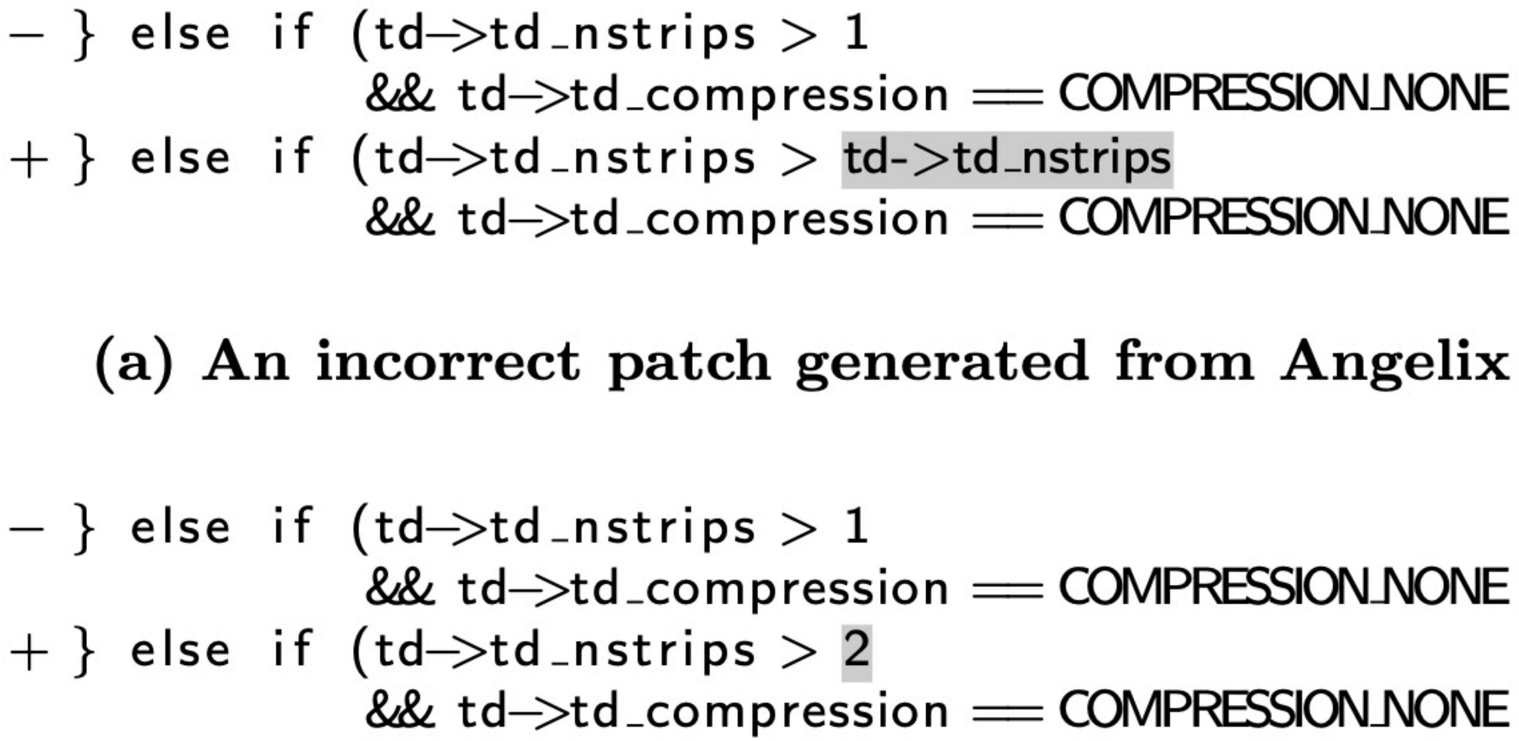


Subject	LoC	Tests	Versions
Wireshark	2814K	63	5
PHP	1046K	85	21
Gzip	491K	12	4
GMP	145K	146	2
Libtiff	77K	78	18

실험 결과


correct patches
10
0
Angelix
FAngelix
w/o Ref
L-37133 (C)
L-3b848 (C)
L-e8a47 (A)
M-13421 (A)
M-14167 (A)
P-63673 (C)
P-8138f (C)
P-e65d3 (C)
Z-3eb60 (C)

실험 결과



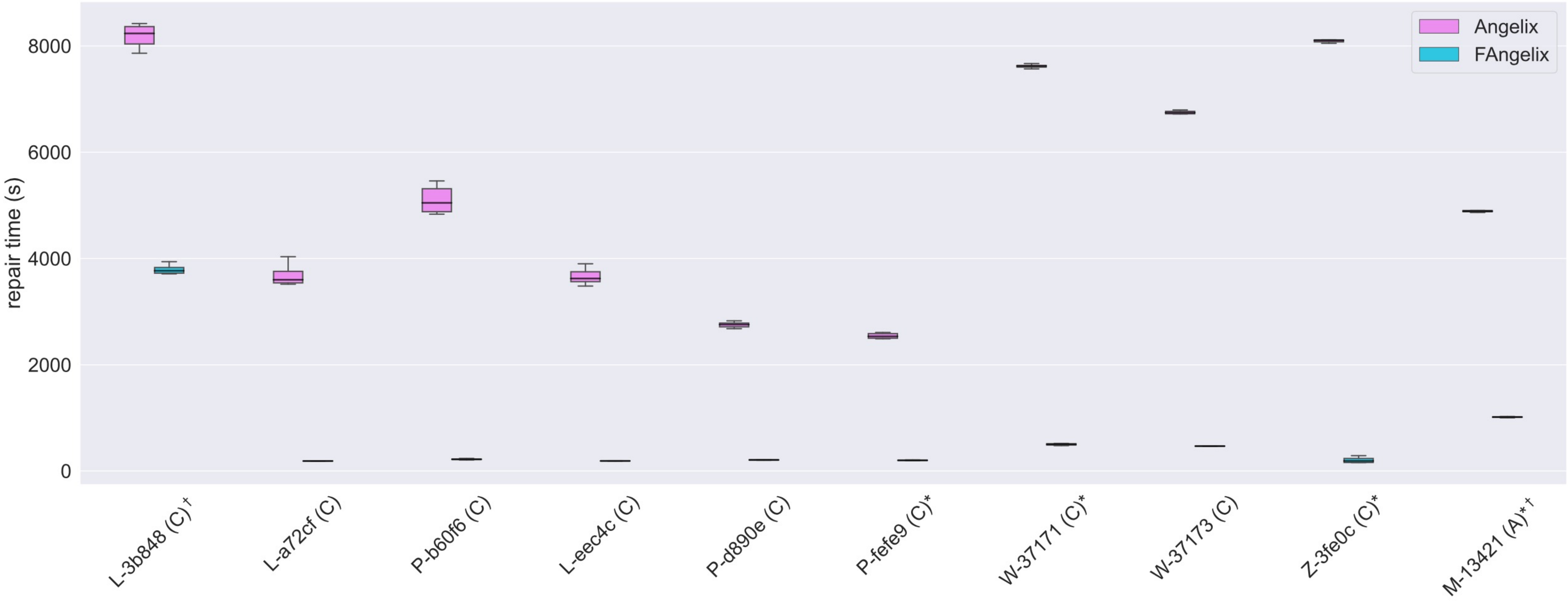
```
− } else if (td−>td_nstrips > 1
                && td−>td_compression == COMPRESSION_NONE
+ } else if (td−>td_nstrips > td->td_nstrips
                && td−>td_compression == COMPRESSION_NONE
```

(a) An incorrect patch generated from Angelix

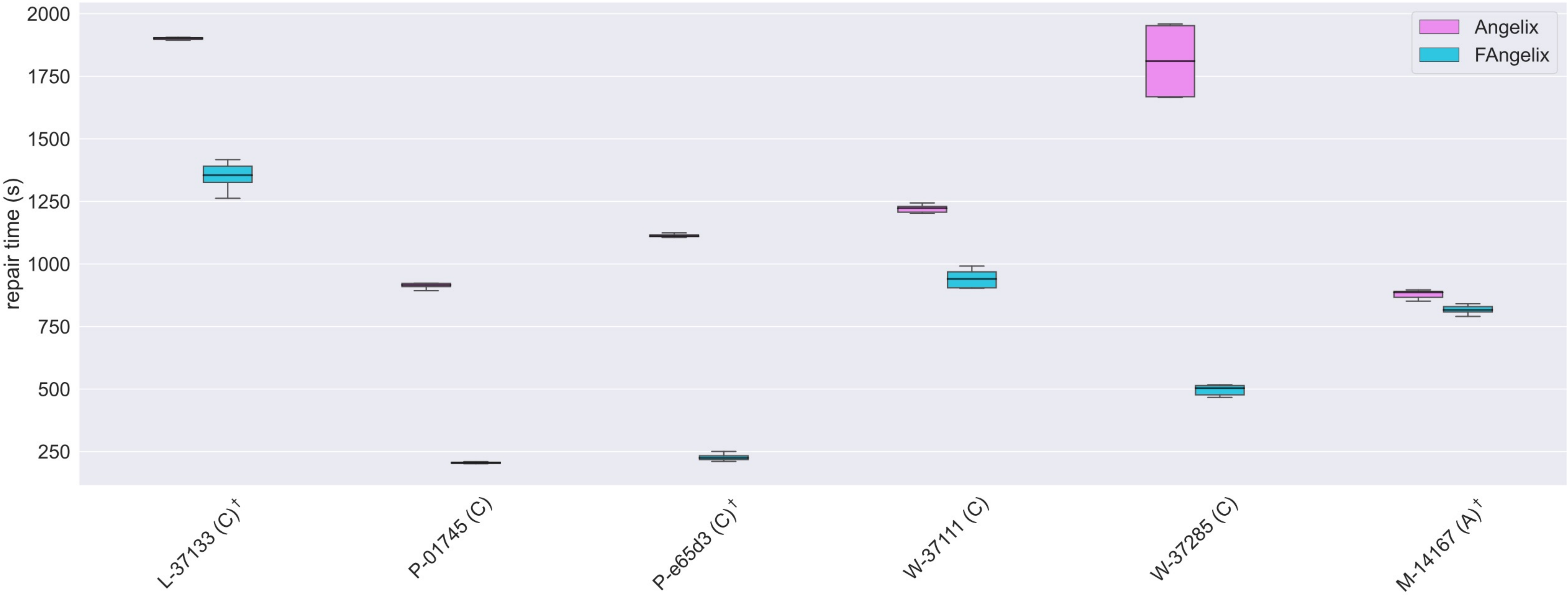
```
− } else if (td−>td_nstrips > 1
                && td−>td_compression == COMPRESSION_NONE
+ } else if (td−>td_nstrips > 2
                && td−>td_compression == COMPRESSION_NONE
```

(b) A correct patch generated from FAngelix

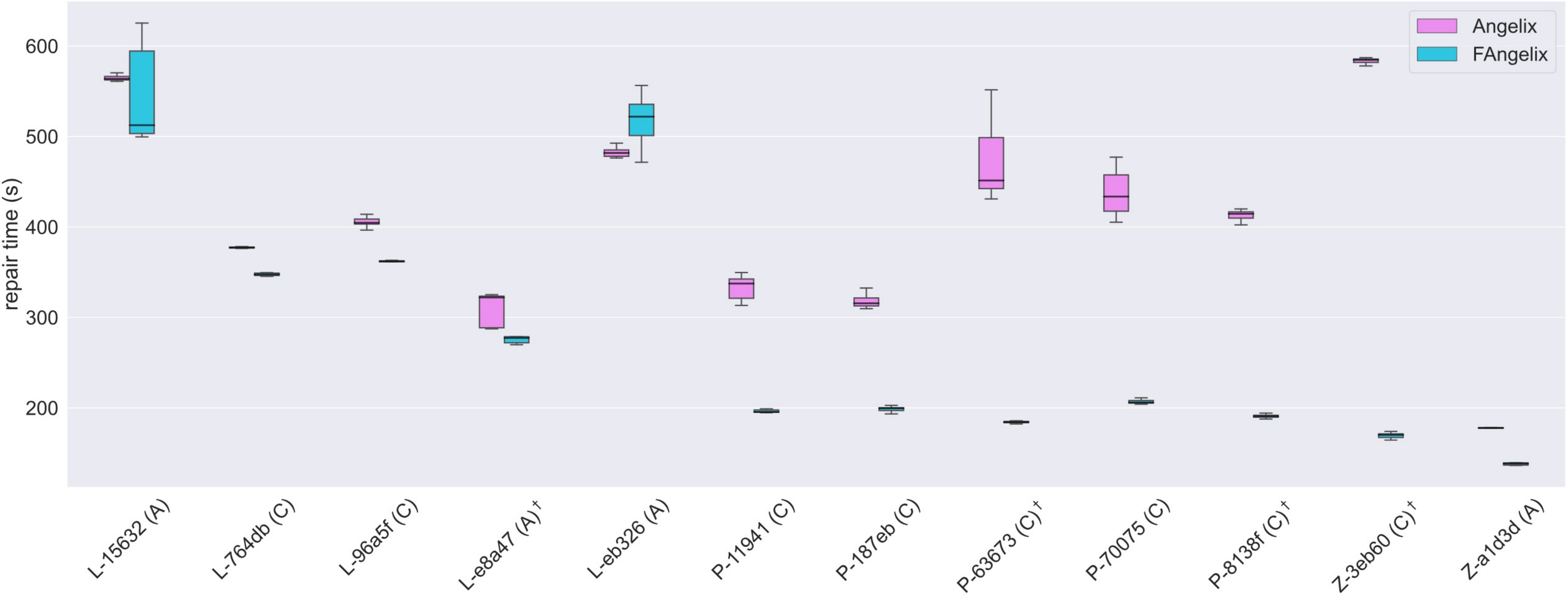
실험 결과 (최대 23배, 평균 3.5배 속도 향상)



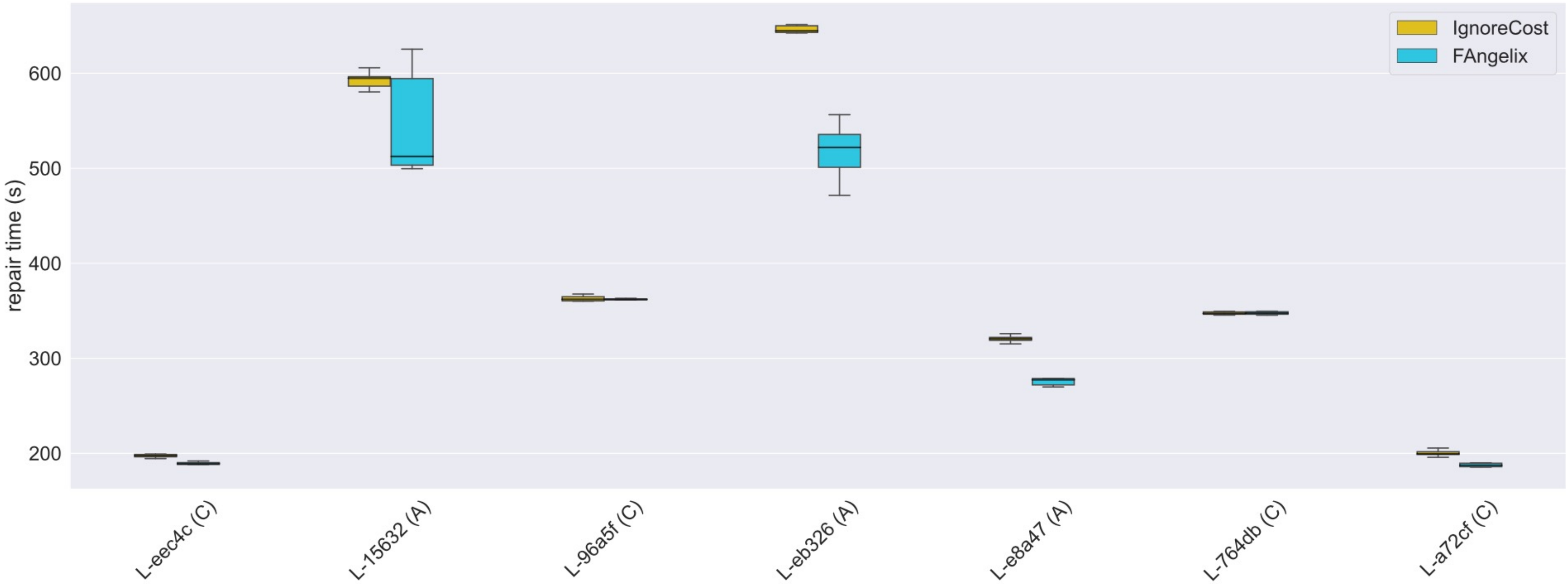
실험 결과 (최대 23배, 평균 3.5배 속도 향상)

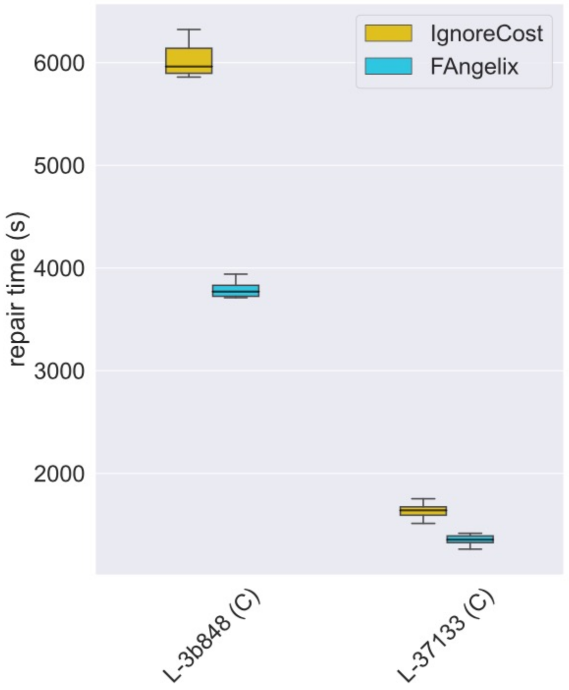


실험 결과 (최대 23배, 평균 3.5배 속도 향상)

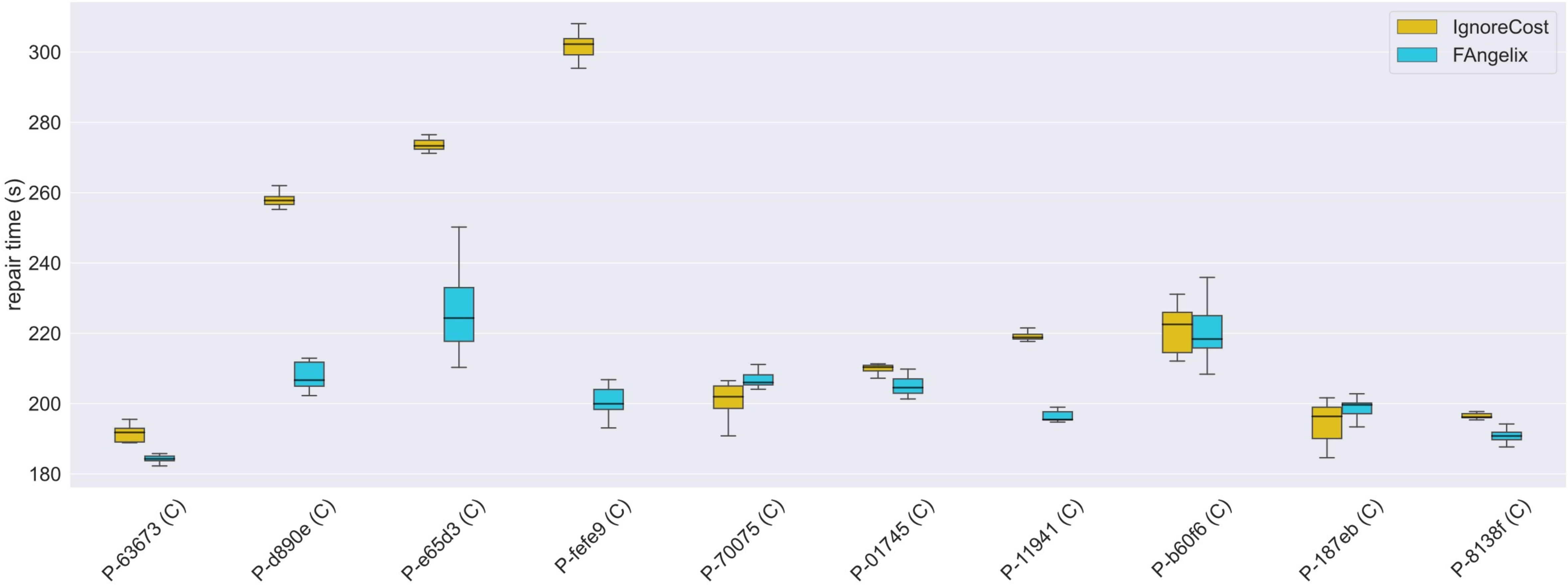


Cost 사용이 효과가 있는가?

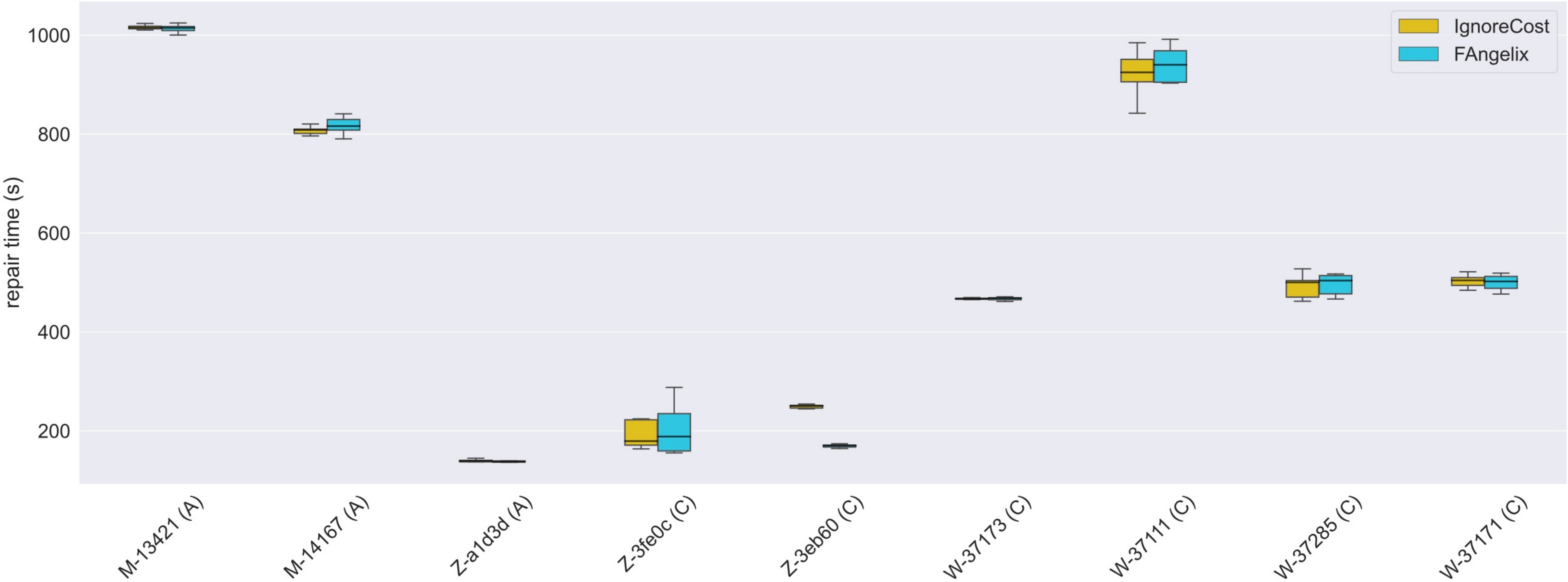




Cost 사용이 효과가 있는가?



Cost 사용이 효과가 있는가?

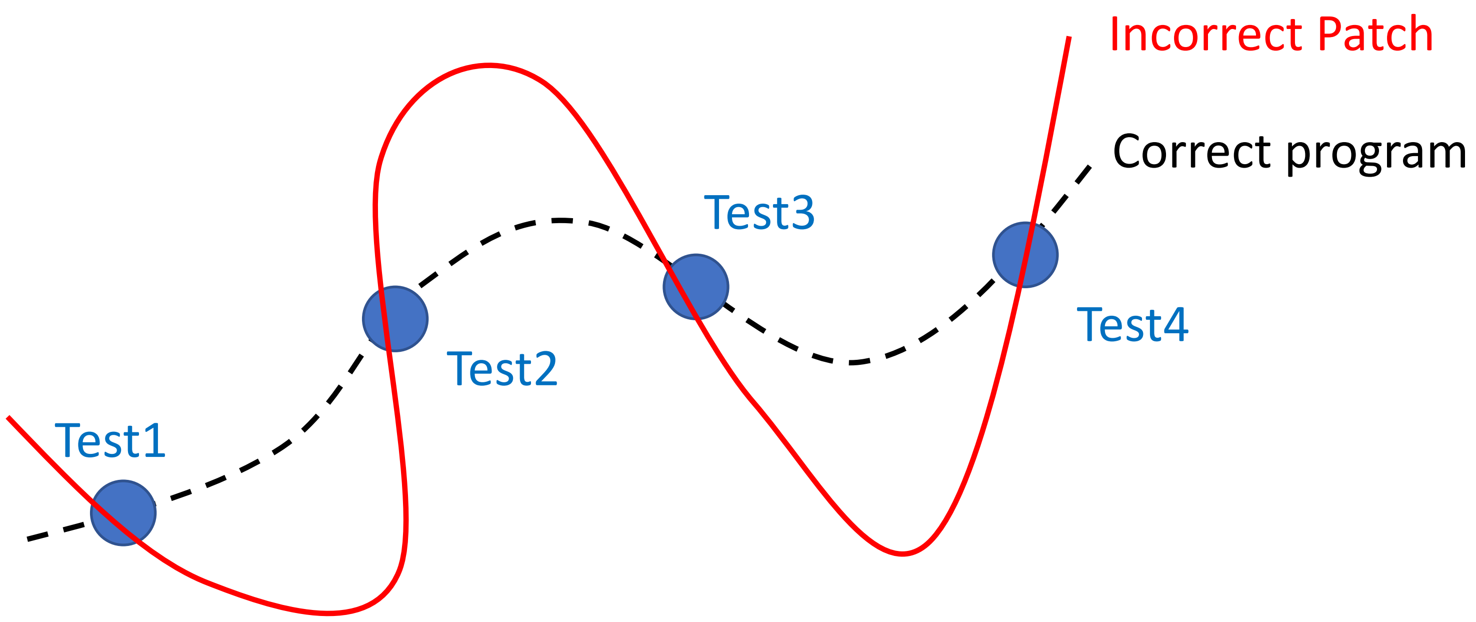


Verifix: Verified Repair of Programming Assignments

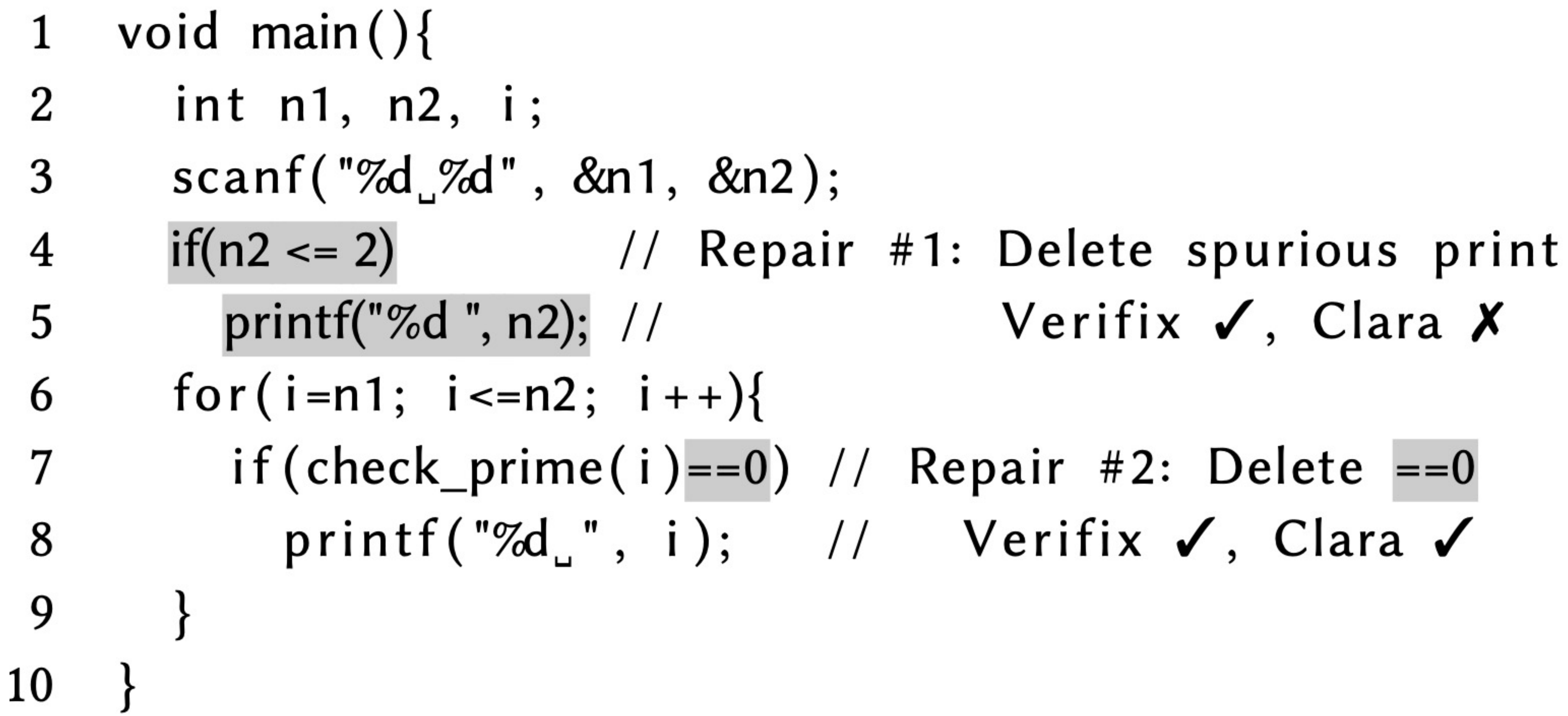
UMAIR Z. AHMED, ZHIYU FAN,
JOOYONG YI, OMAR I. AL-BATAINEH,
ABHIK ROYCHOUDHURY

TOSEM, 2022

Overfitting Problem

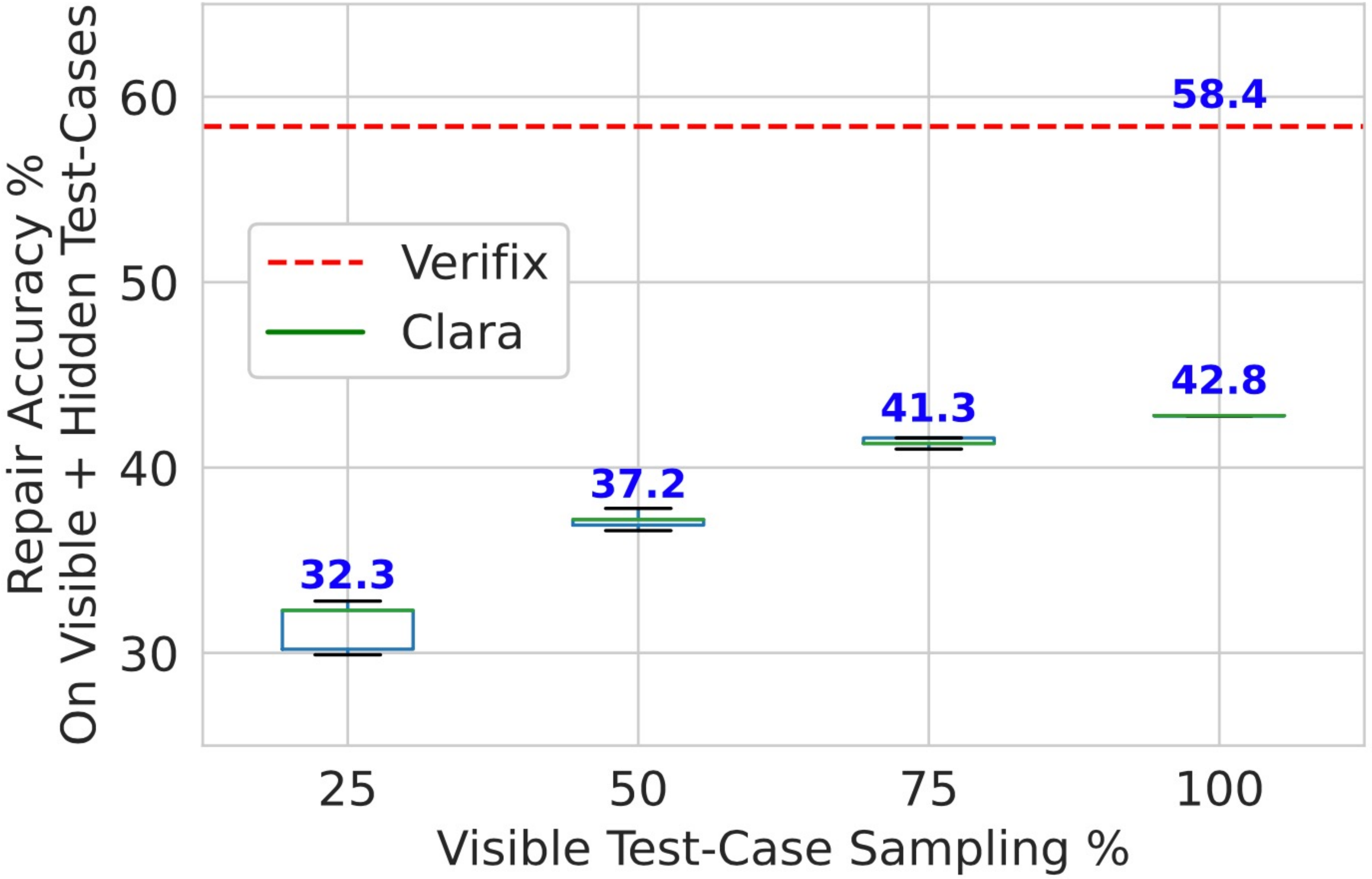


Overfitting Problem

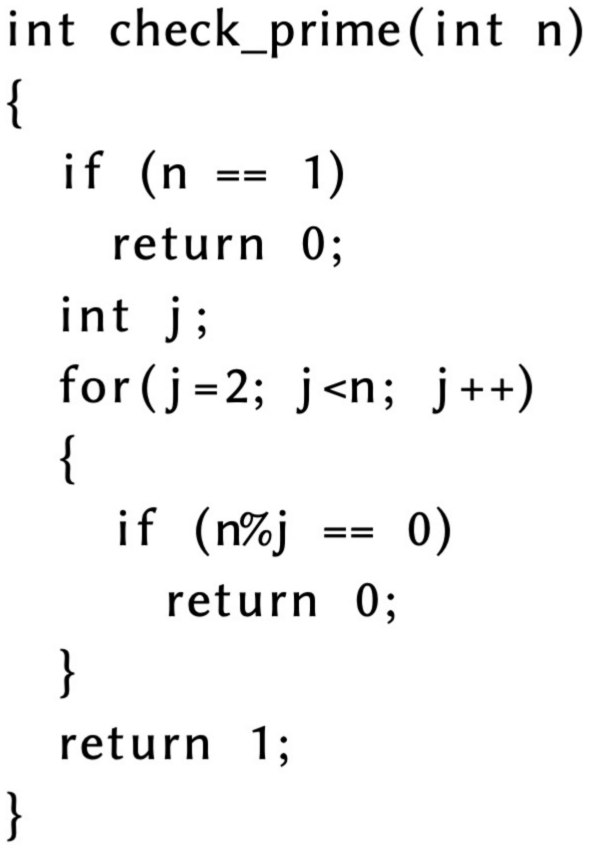


```
void main(){
   int n1, n2, i;
   scanf("%d␣%d", &n1, &n2);
   if(n2 <= 2)                // Repair #1: Delete spurious print
      printf("%d ", n2);     //                  Verifix ✓, Clara ✗
   for(i=n1; i<=n2; i++){
      if(check_prime(i)==0)  // Repair #2: Delete ==0
         printf("%d␣", i);   //    Verifix ✓, Clara ✓
   }
}
```

Overfitting Problem

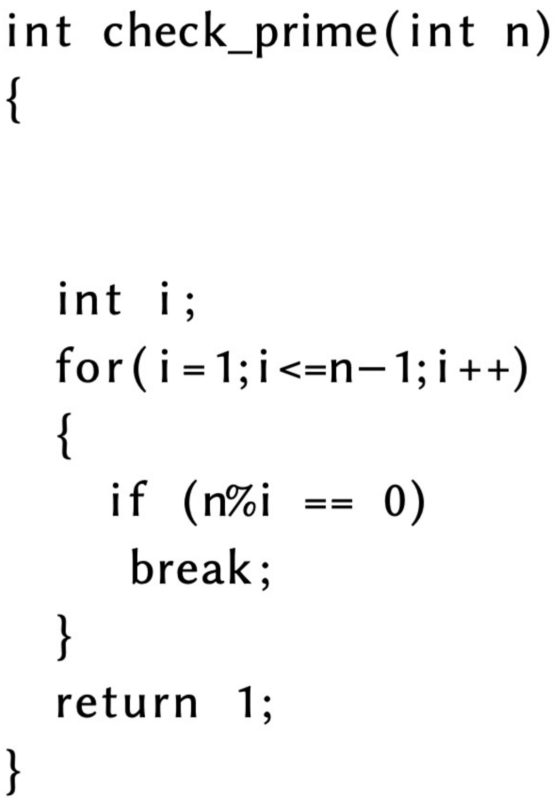


Approach



```
int check_prime(int n)
{
  if (n == 1)
    return 0;
  int j;
  for(j=2; j<n; j++)
  {
    if (n%j == 0)
      return 0;
  }
  return 1;
}
```

Reference program



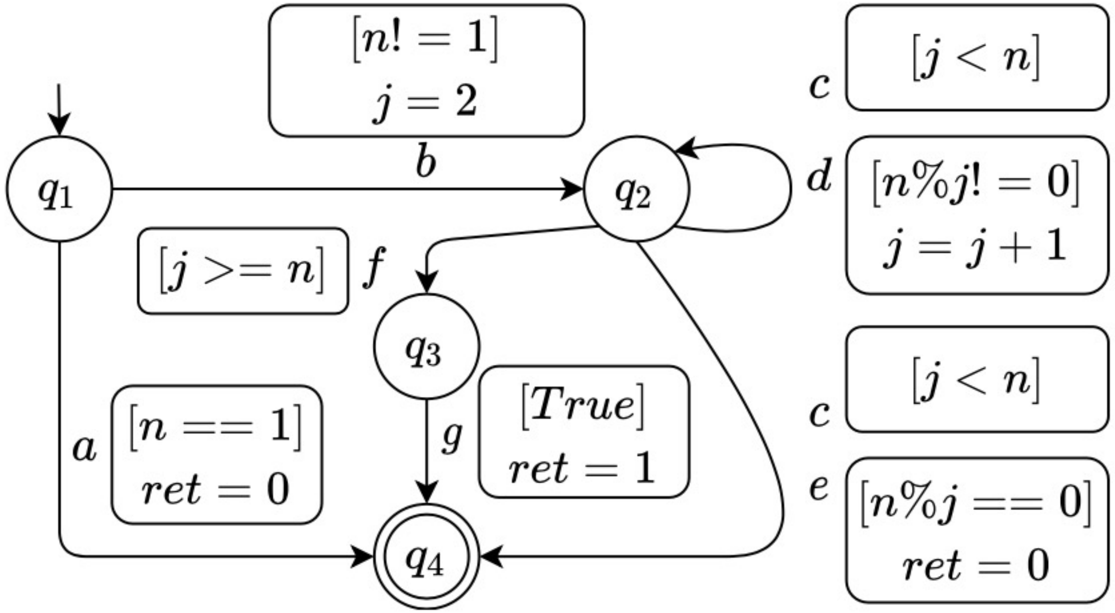
```
int check_prime(int n)
{


  int i;
  for(i=1;i<=n-1;i++)
  {
    if (n%i == 0)
     break;
  }
  return 1;
}
```

Incorrect student program

Program → Control Flow Automata

```
int check_prime(int n)
{
  if (n == 1)
    return 0;
  int j;
  for(j=2; j<n; j++)
  {
    if (n%j == 0)
      return 0;
  }
  return 1;
}
```

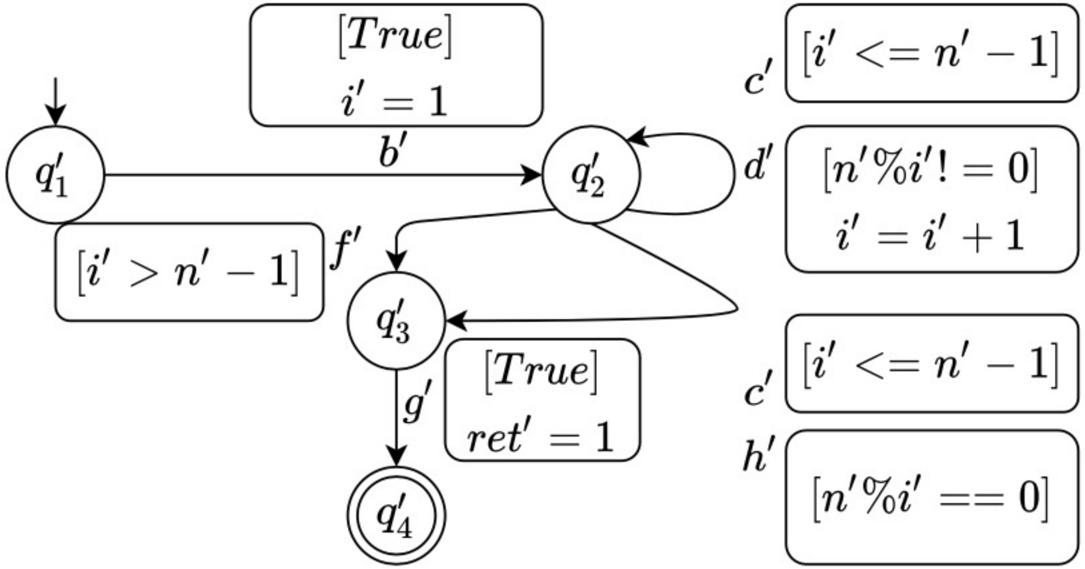
Reference program



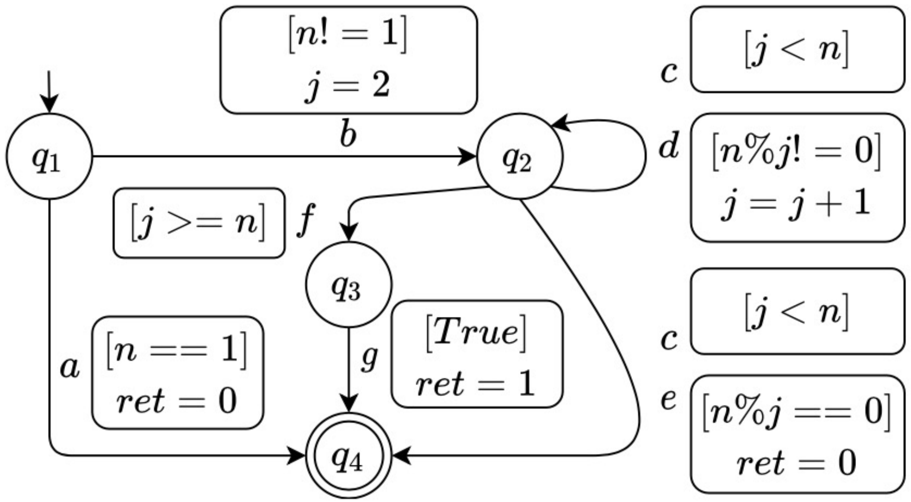
Program → Control Flow Automata

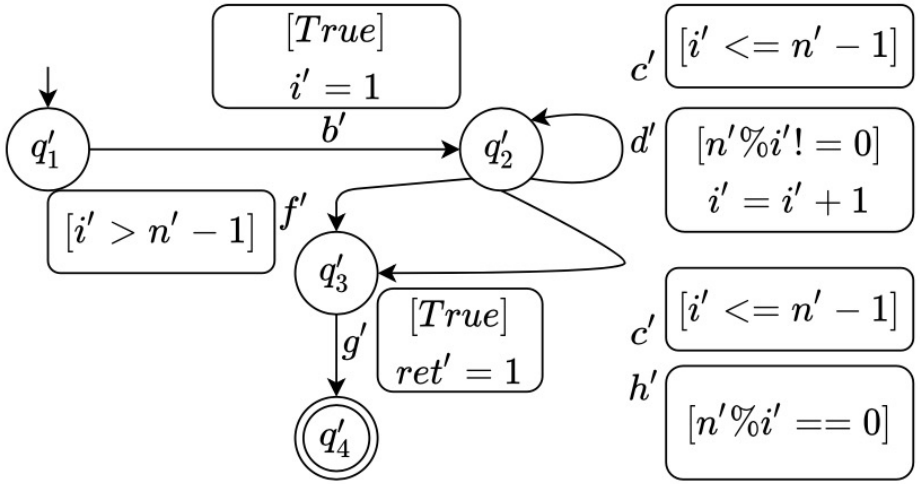
```
int check_prime(int n)
{


   int i;
   for(i=1;i<=n−1;i++)
   {
      if (n%i == 0)
       break;
   }
   return 1;
}
```

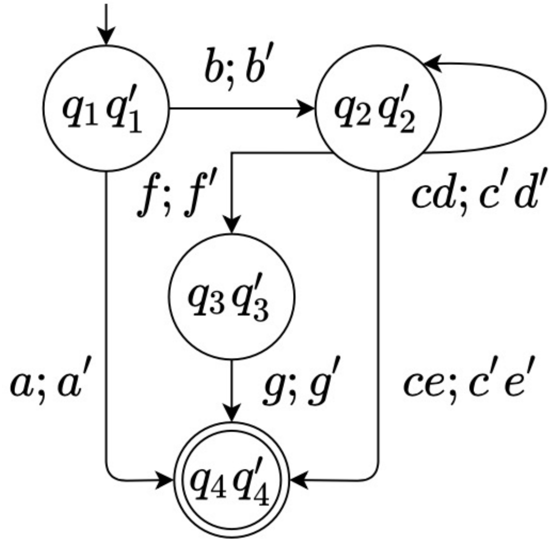
Incorrect student program



Aligning CFAs and variables

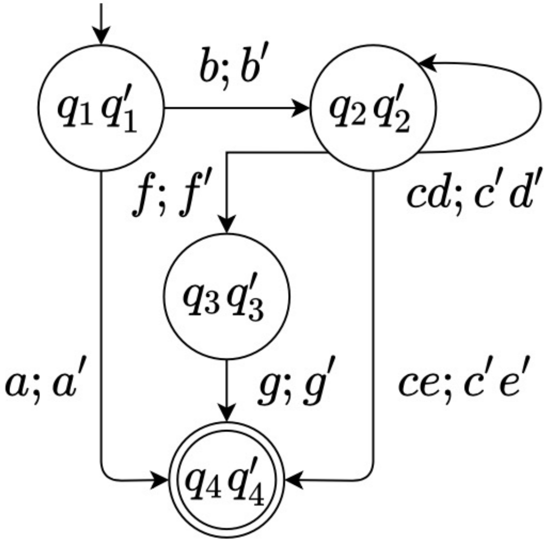






$\{ret \leftrightarrow ret', n \leftrightarrow n', j \leftrightarrow i'\}$

각 edge별로 검증 수행

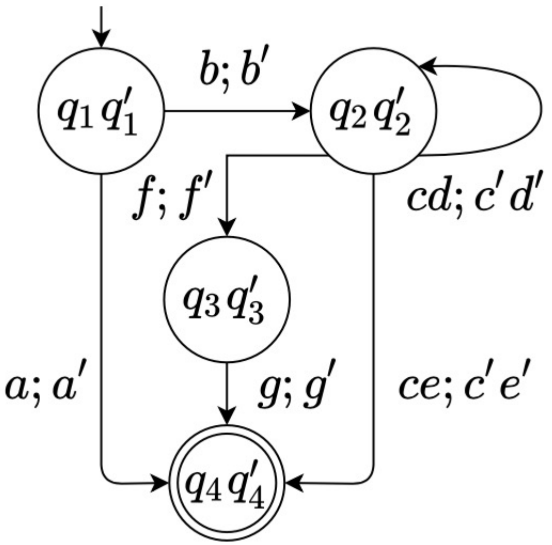


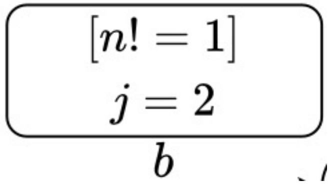
$$\{ret \leftrightarrow ret', n \leftrightarrow n', j \leftrightarrow i'\}$$

$$\varphi^1_{edge} : \phi_{q_1 q'_1} \wedge \psi_r \wedge \psi^1_s \wedge \neg\phi_{q_2 q'_2}$$

$$\phi_{q_1 q'_1}: \quad (ret_0 = ret'_0) \wedge (n_0 = n'_0) \wedge (j_0 = i'_0)$$
$$\phi_{q_2 q'_2}: \quad (ret_1 = ret'_1) \wedge (n_1 = n'_1) \wedge (j_1 = i'_1)$$

각 edge별로 검증 수행

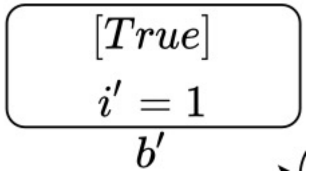




$[n! = 1]$

$j = 2$

b



$[True]$

$i' = 1$

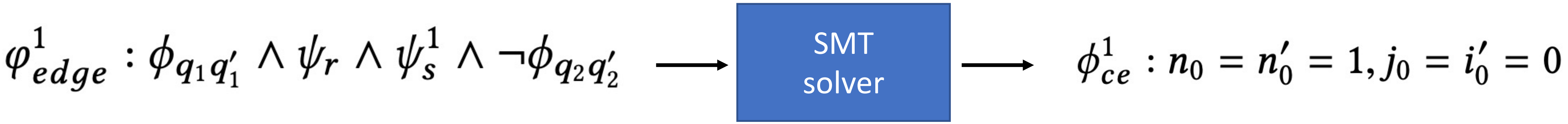
b'

$$\varphi^1_{edge} : \phi_{q_1 q'_1} \land \psi_r \land \psi^1_s \land \neg\phi_{q_2 q'_2}$$

$$\psi_r: \quad (n_0 \neq 1 \implies j_1 = 2) \quad \land \quad (\neg(n_0 \neq 1) \implies j_1 = j_0)$$

$$\psi^1_s: \quad (True \implies i'_1 = 1) \quad \land \quad (\neg True \implies i'_1 = i'_0)$$

Minimal Edge Repair via CEGIS



$\varphi^1_{edge} : \phi_{q_1 q'_1} \wedge \psi_r \wedge \psi^1_s \wedge \neg\phi_{q_2 q'_2}$

SMT solver

$\phi^1_{ce} : n_0 = n'_0 = 1, j_0 = i'_0 = 0$

Minimal Edge Repair via CEGIS

$$\varphi^1_{edge} : \phi_{q_1 q_1'} \wedge \psi_r \wedge \psi_s^1 \wedge \neg\phi_{q_2 q_2'} \longrightarrow \boxed{\text{SMT solver}} \longrightarrow \phi^1_{ce} : n_0 = n_0' = 1, j_0 = i_0' = 0$$

$$\phi_{q_1 q_1'} \wedge \psi_r \wedge \cancel{\psi_s^1} \wedge \neg\phi_{q_2 q_2'} \wedge \phi^1_{ce} \longrightarrow \boxed{\text{SMT solver}} \longrightarrow \text{Unsat}$$

$$\uparrow$$

$$(n_0' \neq 1 \implies i_1' = 1) \quad \wedge \quad (\neg(n_0' \neq 1) \implies i_1' = i_0')$$

Minimal Edge Repair via CEGIS

$$\varphi^1_{edge} : \phi_{q_1q_1'} \wedge \psi_r \wedge \psi^1_s \wedge \neg\phi_{q_2q_2'} \longrightarrow \boxed{\begin{array}{c}\text{SMT}\\ \text{solver}\end{array}} \longrightarrow \phi^1_{ce} : n_0 = n_0' = 1, j_0 = i_0' = 0$$

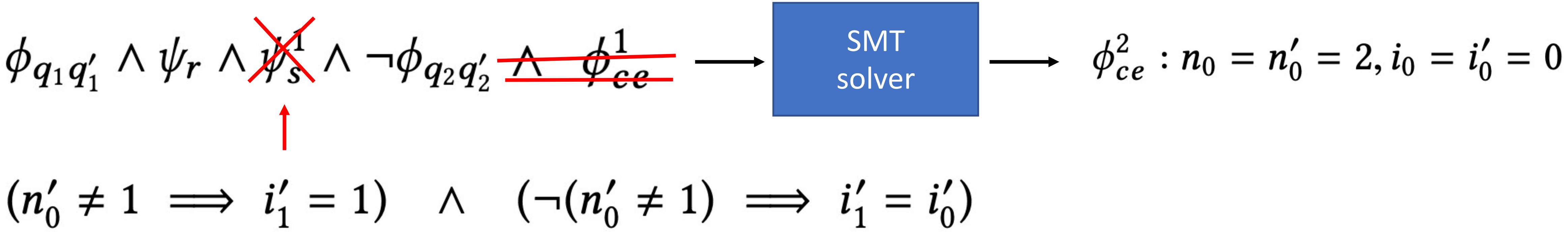
$$\phi_{q_1q_1'} \wedge \psi_r \wedge \cancel{\psi^1_s} \wedge \neg\phi_{q_2q_2'} \ \wedge\ \phi^1_{ce} \longrightarrow \boxed{\begin{array}{c}\text{SMT}\\ \text{solver}\end{array}} \longrightarrow \text{Unsat}$$

$$\uparrow$$

$$(n_0' \neq 1 \implies i_1' = 1) \quad \wedge \quad (\neg(n_0' \neq 1) \implies i_1' = i_0')$$

$$\begin{aligned} \psi_r&: \quad (n_0 \neq 1 \implies j_1 = 2) \quad \wedge \quad (\neg(n_0 \neq 1) \implies j_1 = j_0) \\ \psi^1_s&: \quad (True \implies i_1' = 1) \quad \wedge \quad (\neg True \implies i_1' = i_0') \end{aligned}$$

Minimal Edge Repair via CEGIS



$$\phi_{q_1 q_1'} \wedge \psi_r \wedge \xcancel{\psi_s^1} \wedge \neg\phi_{q_2 q_2'} \cancel{\wedge \quad \phi_{ce}^1} \longrightarrow \boxed{\text{SMT solver}} \longrightarrow \phi_{ce}^2 : n_0 = n_0' = 2, i_0 = i_0' = 0$$

$$\uparrow$$

$$(n_0' \neq 1 \implies i_1' = 1) \quad \wedge \quad (\neg(n_0' \neq 1) \implies i_1' = i_0')$$

Minimal Edge Repair via CEGIS

$$\phi_{q_1q_1'} \wedge \psi_r \wedge \underset{\uparrow}{\cancel{\psi_s^1}} \wedge \neg\phi_{q_2q_2'} \;\wedge\; \phi_{ce}^1 \;\wedge\; \phi_{ce}^2 \longrightarrow \boxed{\begin{array}{c}\text{SMT}\\ \text{solver}\end{array}} \longrightarrow \text{Unsat}$$

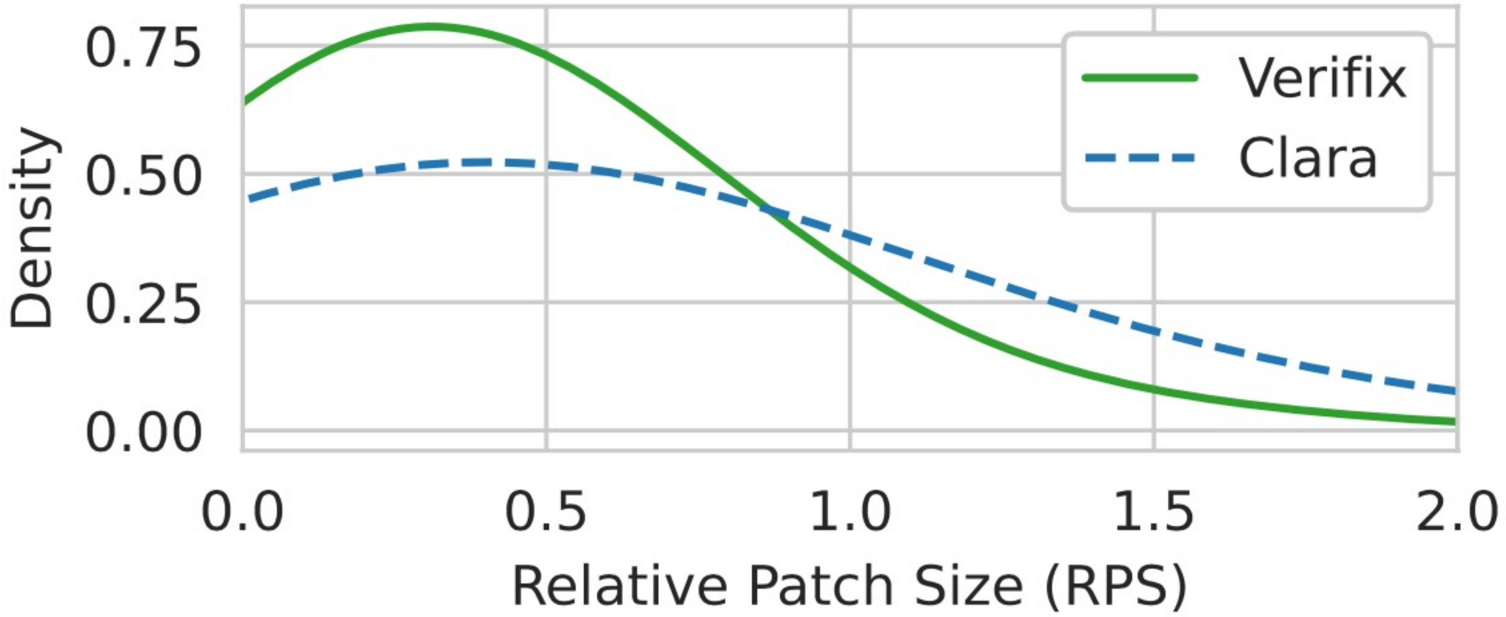
$$(n_0' \neq 1 \implies i_1' = 2) \quad \wedge \quad (\neg(n_0' \neq 1) \implies i_1' = i_0')$$

$$\begin{array}{llll}\psi_r: & (n_0 \neq 1 \implies j_1 = 2) & \wedge & (\neg(n_0 \neq 1) \implies j_1 = j_0) \\ \psi_s^1: & (True \implies i_1' = 1) & \wedge & (\neg True \implies i_1' = i_0')\end{array}$$

실험

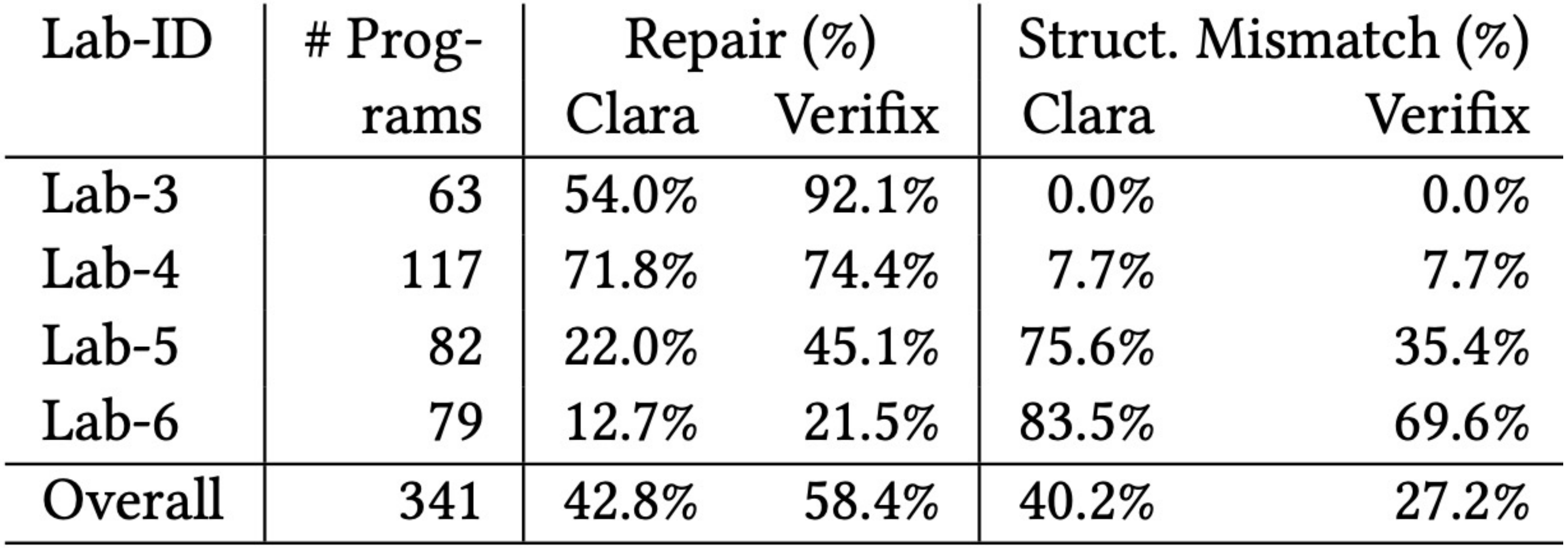
- 실험 대상:
    - 28개의 프로그래밍 문제로부터 취합한 341개의 컴파일 가능한 학생 프로그램
    - 이전 연구[FSE’17]에서 취합한 데이터 (Indian Institute of Technology Kanpur)
    - 각 문제마다 instructor가 작성한 정답 프로그램과 테스트 케이스 존재
    - 비교 대상: Clara [PLDI’18]
        - C 프로그램을 지원하는 최신의 공개된 도구

패치 크기 비교



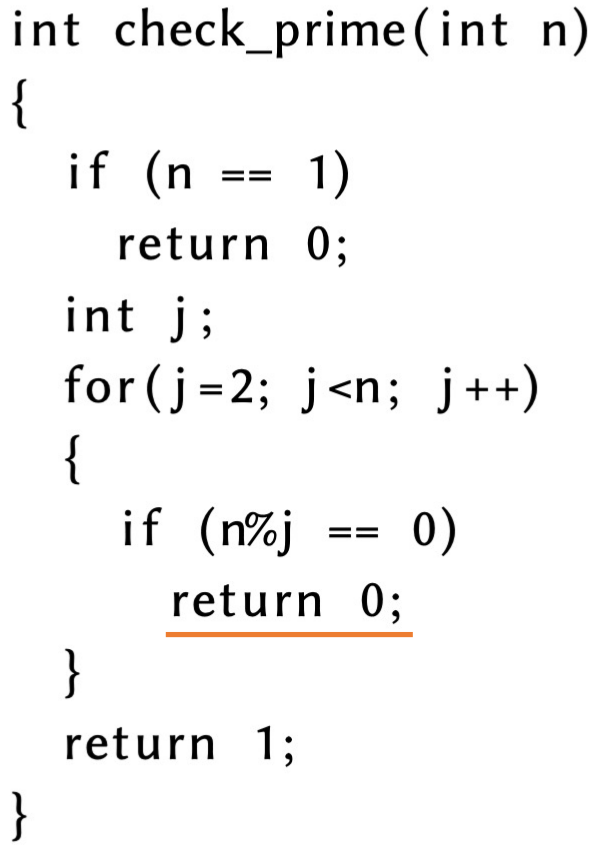
$$RPS = Dist(AST_s, AST_f)/Size(AST_s)$$

패치 성공률 비교 (단일한 정답 프로그램 사용시)



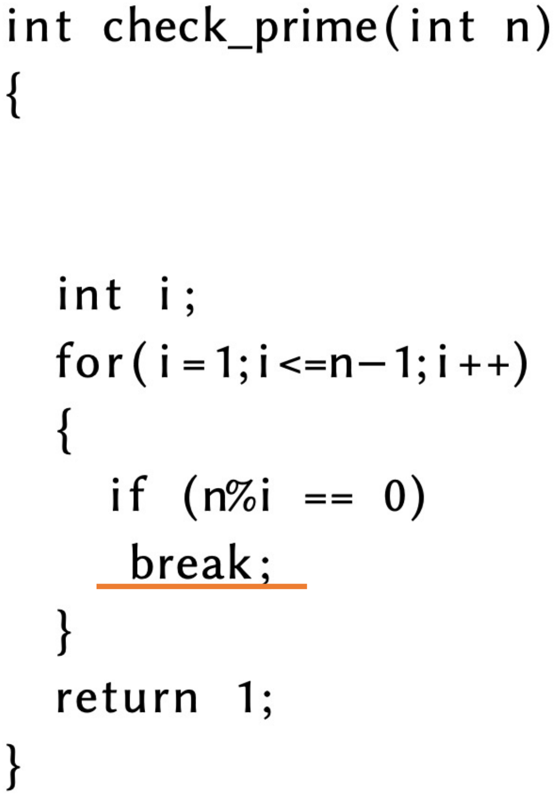
Lab-ID	# Prog-rams	Repair (%)		Struct. Mismatch (%)	
		Clara	Verifix	Clara	Verifix
Lab-3	63	54.0%	92.1%	0.0%	0.0%
Lab-4	117	71.8%	74.4%	7.7%	7.7%
Lab-5	82	22.0%	45.1%	75.6%	35.4%
Lab-6	79	12.7%	21.5%	83.5%	69.6%
Overall	341	42.8%	58.4%	40.2%	27.2%

Clara and Sarfgen fail to handle our example



```
int check_prime(int n)
{
  if (n == 1)
    return 0;
  int j;
  for(j=2; j<n; j++)
  {
    if (n%j == 0)
      return 0;
  }
  return 1;
}
```

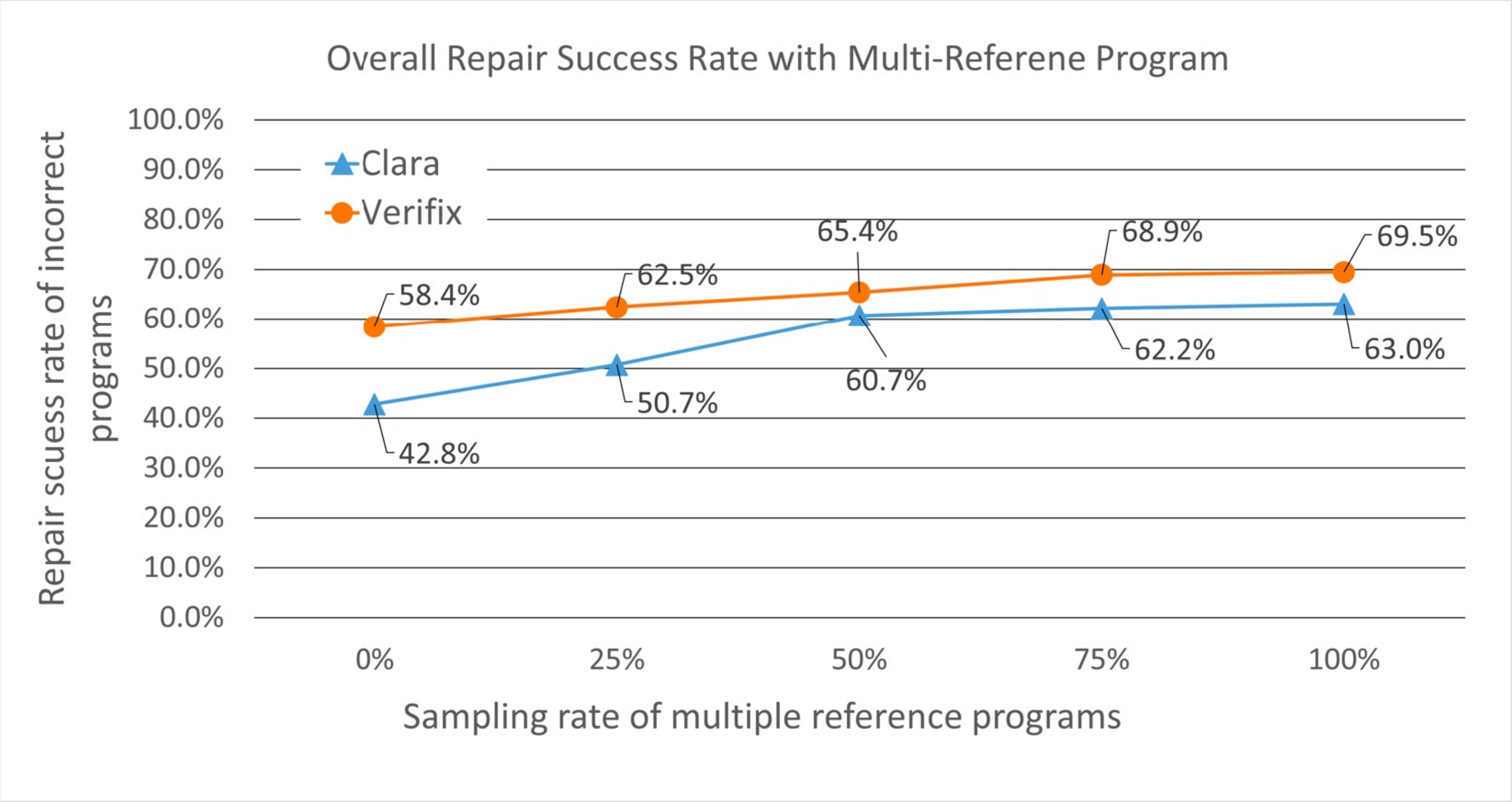
Reference program



```
int check_prime(int n)
{


  int i;
  for(i=1;i<=n-1;i++)
  {
    if (n%i == 0)
     break;
  }
  return 1;
}
```

Incorrect student program

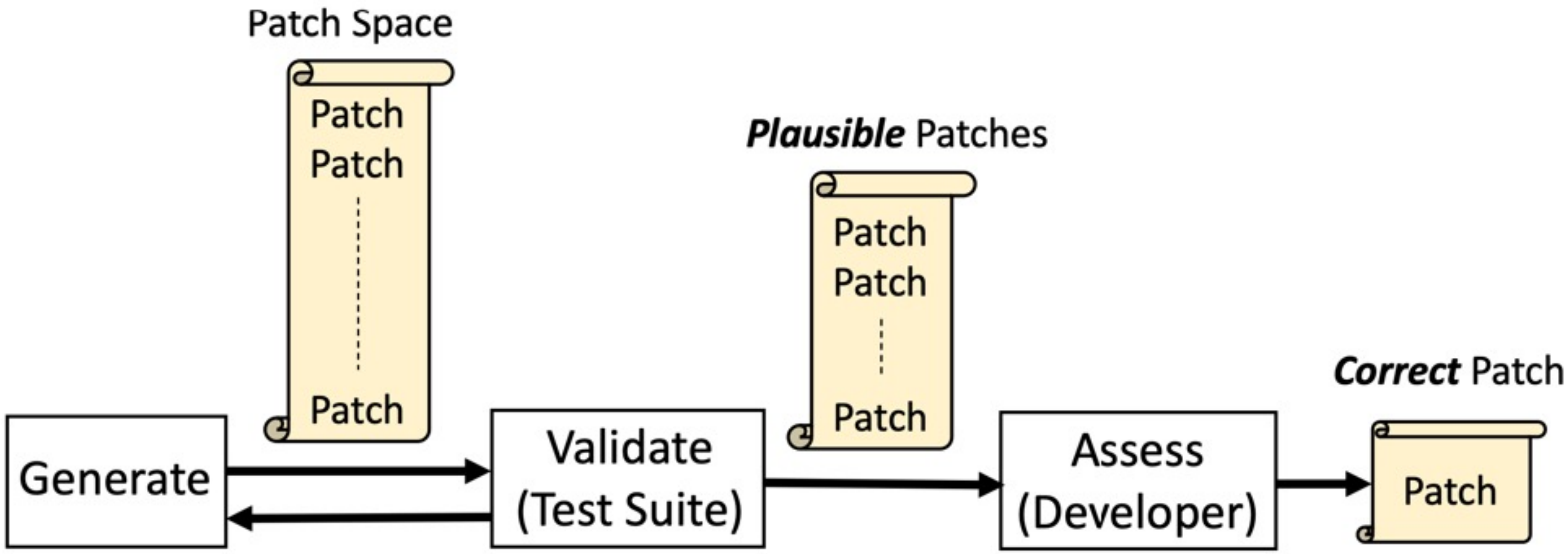
패치 성공률 비교 (다수 정답 프로그램 사용시)



Testing Patches Under Preservation Conditions To Combat the Overfitting Problem of Program Repair

Elkhan Ismayilzada, Md Mazba Ur Rahman,
Dongsun Kim, Jooyong Yi

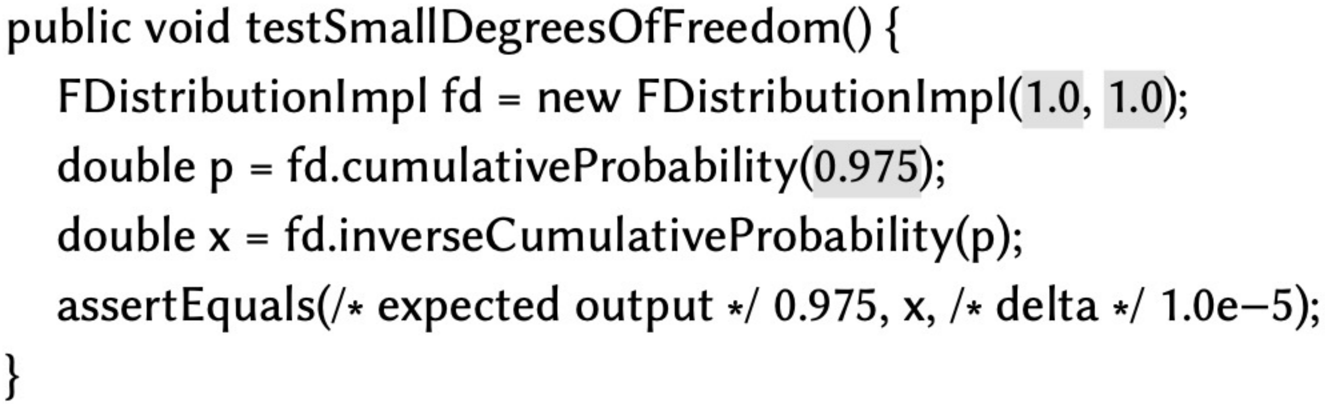
다수의 Plausible 패치 존재



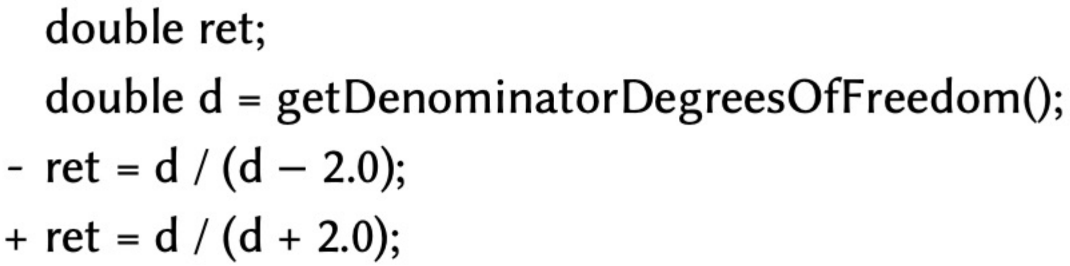
Patch Classification 문제

- 생성된 plausible patch가 correct한 패치인가?

Patch Classification 문제 예

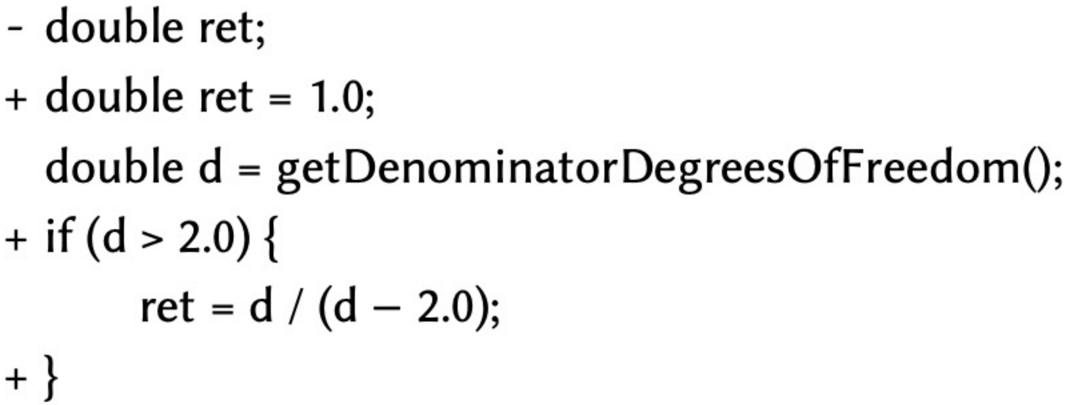


```
public void testSmallDegreesOfFreedom() {
    FDistributionImpl fd = new FDistributionImpl(1.0, 1.0);
    double p = fd.cumulativeProbability(0.975);
    double x = fd.inverseCumulativeProbability(p);
    assertEquals(/* expected output */ 0.975, x, /* delta */ 1.0e−5);
}
```



```
  double ret;
  double d = getDenominatorDegreesOfFreedom();
- ret = d / (d − 2.0);
+ ret = d / (d + 2.0);
```

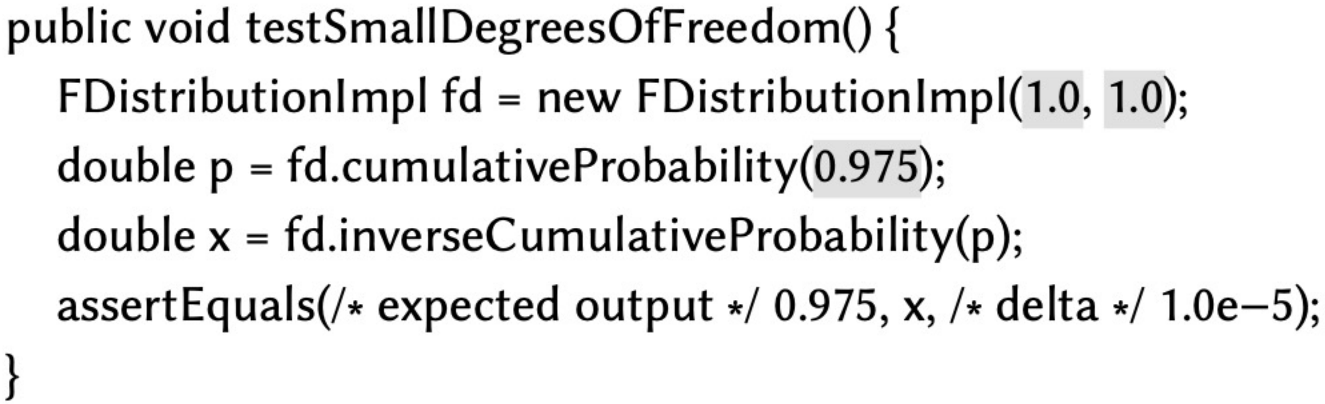
(a) An incorrect patch for Math95



```
- double ret;
+ double ret = 1.0;
  double d = getDenominatorDegreesOfFreedom();
+ if (d > 2.0) {
        ret = d / (d − 2.0);
+ }
```

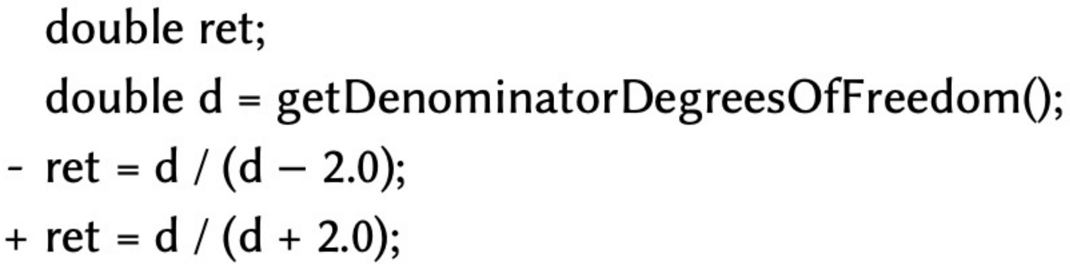
(b) A correct patch for Math95.

Patch Classification 문제 예



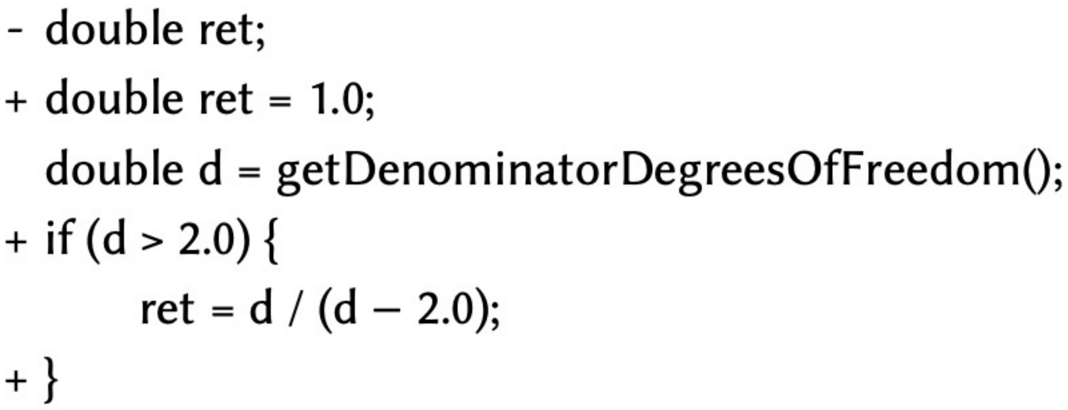
```
public void testSmallDegreesOfFreedom() {
    FDistributionImpl fd = new FDistributionImpl(1.0, 1.0);
    double p = fd.cumulativeProbability(0.975);
    double x = fd.inverseCumulativeProbability(p);
    assertEquals(/* expected output */ 0.975, x, /* delta */ 1.0e−5);
}
```

PATCH-SIM 적용: 모든 (14) incorrect patch를 올바른 패치로 인식



```
  double ret;
  double d = getDenominatorDegreesOfFreedom();
- ret = d / (d − 2.0);
+ ret = d / (d + 2.0);
```

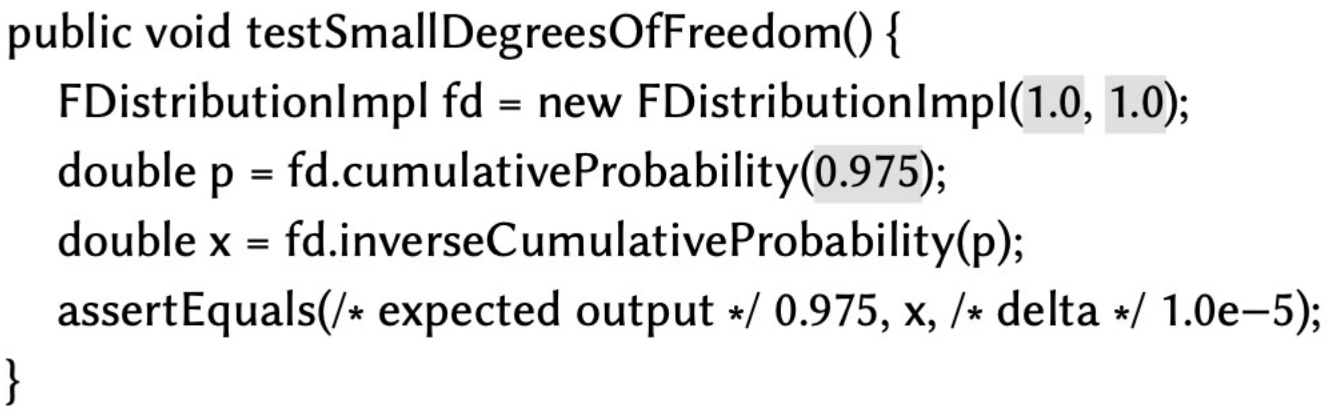
(a) An incorrect patch for Math95



```
- double ret;
+ double ret = 1.0;
  double d = getDenominatorDegreesOfFreedom();
+ if (d > 2.0) {
        ret = d / (d − 2.0);
+ }
```

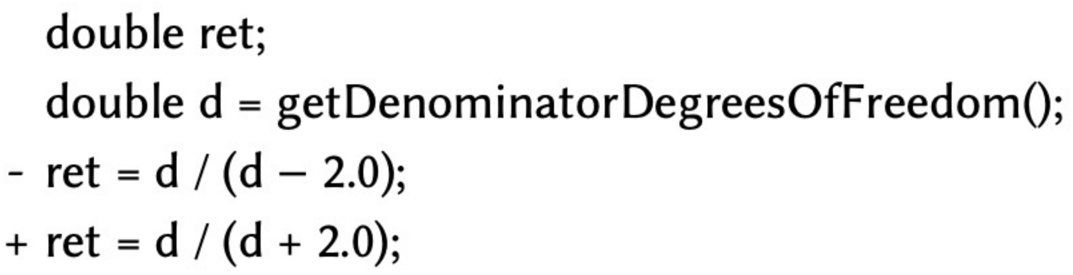
(b) A correct patch for Math95.

Patch Classification 문제 예



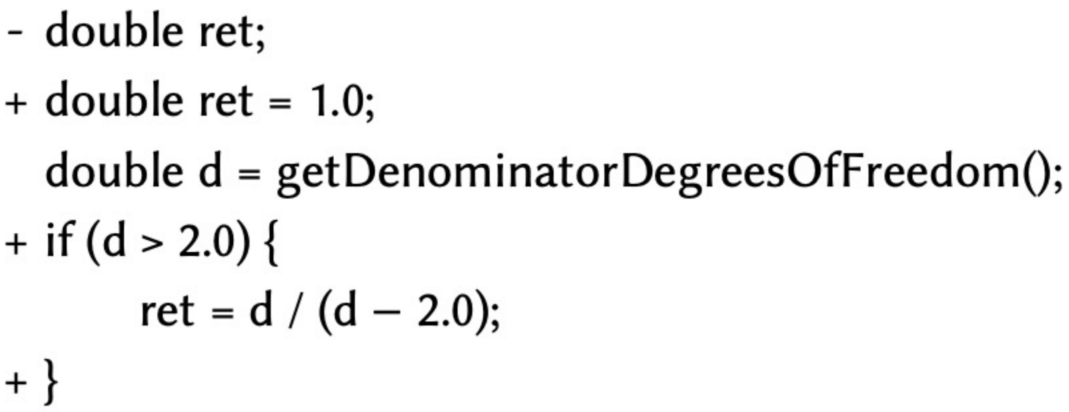
```
public void testSmallDegreesOfFreedom() {
    FDistributionImpl fd = new FDistributionImpl(1.0, 1.0);
    double p = fd.cumulativeProbability(0.975);
    double x = fd.inverseCumulativeProbability(p);
    assertEquals(/* expected output */ 0.975, x, /* delta */ 1.0e−5);
}
```

ODS 적용: 올바른 패치를 그릇된 패치로 인식



```
   double ret;
   double d = getDenominatorDegreesOfFreedom();
-  ret = d / (d − 2.0);
+  ret = d / (d + 2.0);
```

(a) An incorrect patch for Math95



```
-  double ret;
+  double ret = 1.0;
   double d = getDenominatorDegreesOfFreedom();
+  if (d > 2.0) {
         ret = d / (d − 2.0);
+  }
```

(b) A correct patch for Math95.

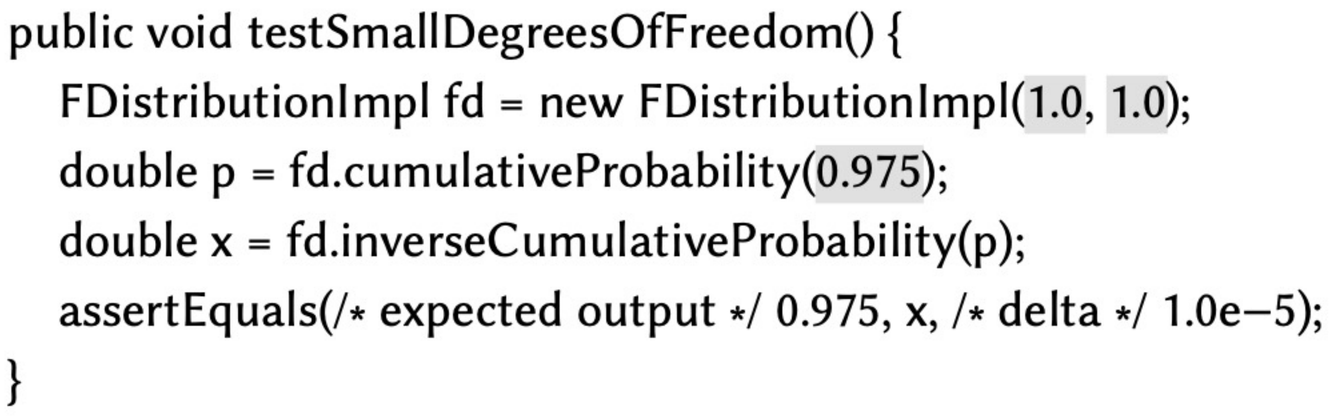
Patch Classification의 어려움

- Score-based 접근법:
  - recall과 precision을 모두 높이도록 임계값을 설정하기 어려움

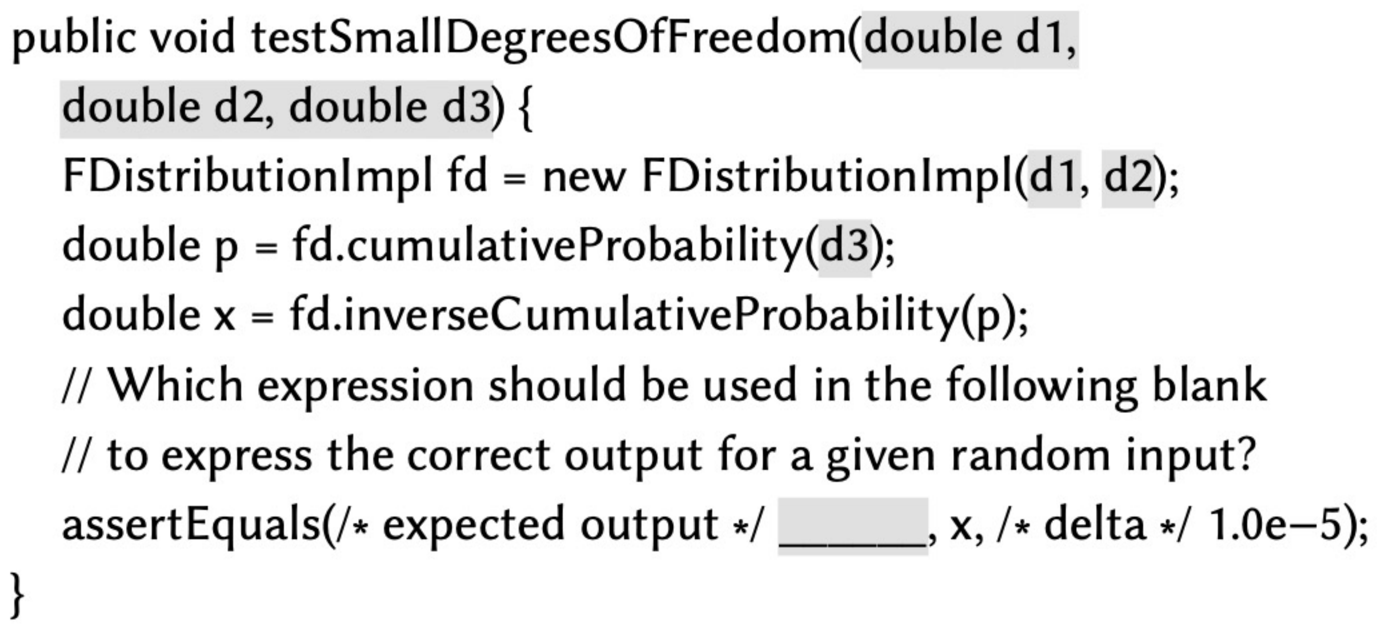
Patch Classification의 어려움

- Evidence-based 접근법:
    - 패치된 프로그램이 새 입력값에 대해 crash를 일으키면 고려 대상에서 제외
        - 일반적으로 적용하기 어려움
        - Java 등의 언어에서는 exception 생성이 오히려 기대되는 테스트도 존재

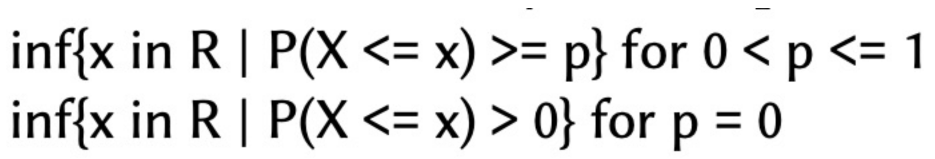
근본 문제: 정확한 명세의 부재



```
public void testSmallDegreesOfFreedom() {
   FDistributionImpl fd = new FDistributionImpl(1.0, 1.0);
   double p = fd.cumulativeProbability(0.975);
   double x = fd.inverseCumulativeProbability(p);
   assertEquals(/* expected output */ 0.975, x, /* delta */ 1.0e−5);
}
```



```
public void testSmallDegreesOfFreedom(double d1,
   double d2, double d3) {
   FDistributionImpl fd = new FDistributionImpl(d1, d2);
   double p = fd.cumulativeProbability(d3);
   double x = fd.inverseCumulativeProbability(p);
   // Which expression should be used in the following blank
   // to express the correct output for a given random input?
   assertEquals(/* expected output */ ______, x, /* delta */ 1.0e−5);
}
```



inf{x in R | P(X <= x) >= p} for 0 < p <= 1

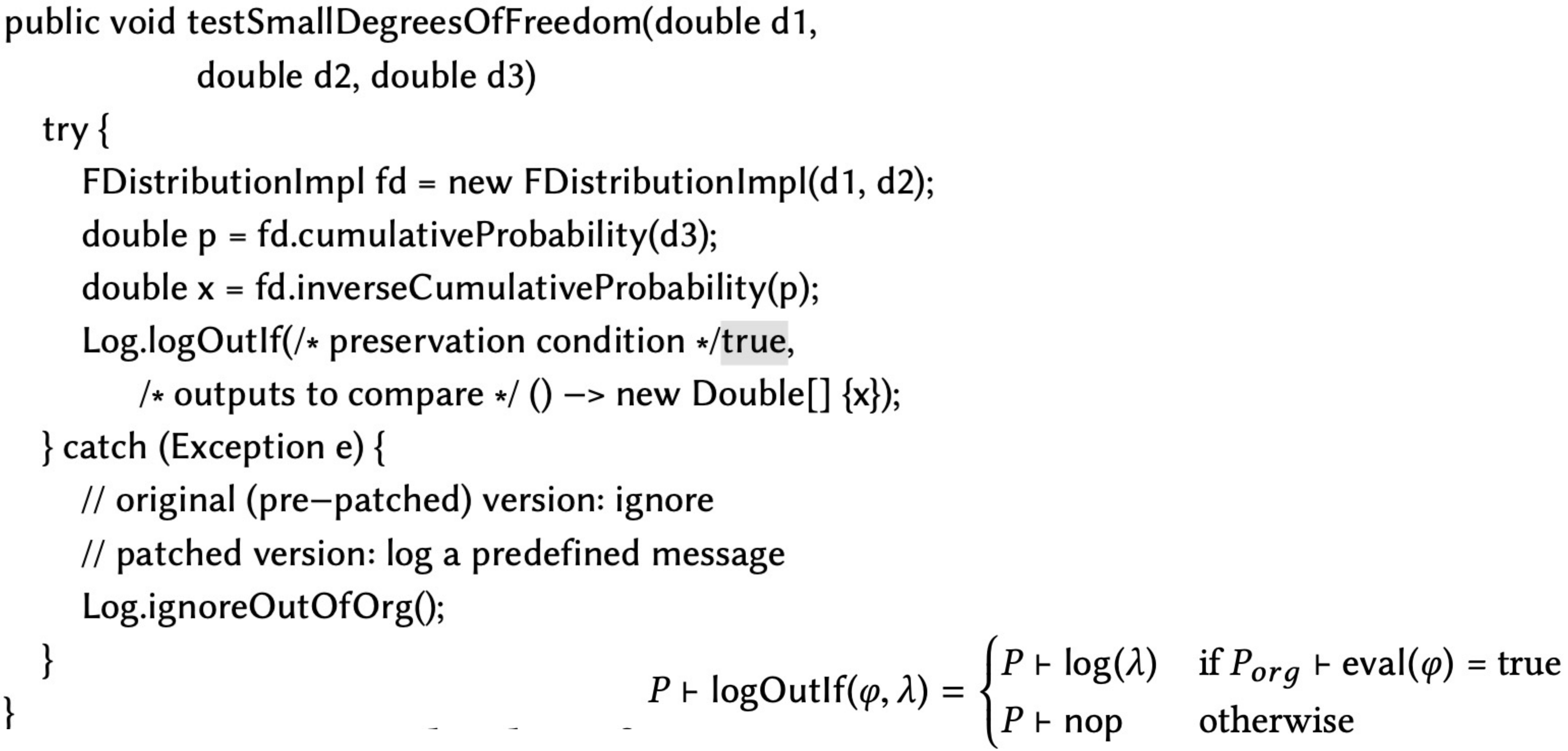
inf{x in R | P(X <= x) > 0} for p = 0

Change Contract [ISSTA’13]

Stack.scc

```
public class Stack<E> {

/*@changed_behavior
  @ when_signaled (IllegalStateException e)
  @  e.getErrorCode() == 100; // observed erroneous behavior
  @ signals (IllegalStateException e) false; // should be fixed
  @*/
  public void push(E item);
}
```

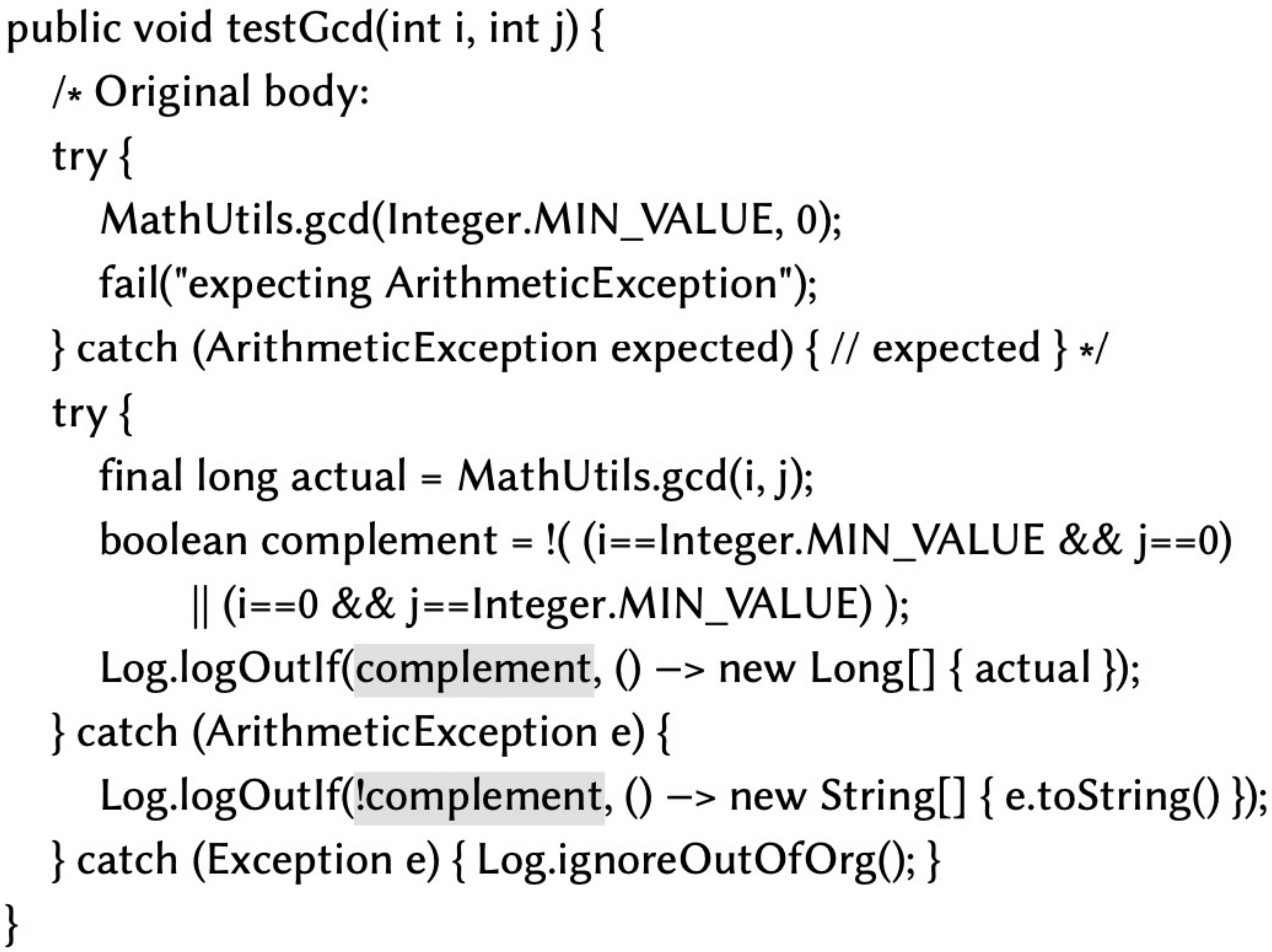
Preservation Condition



```
public void testSmallDegreesOfFreedom(double d1,
                double d2, double d3)
   try {
      FDistributionImpl fd = new FDistributionImpl(d1, d2);
      double p = fd.cumulativeProbability(d3);
      double x = fd.inverseCumulativeProbability(p);
      Log.logOutIf(/* preservation condition */true,
          /* outputs to compare */ () -> new Double[] {x});
   } catch (Exception e) {
      // original (pre-patched) version: ignore
      // patched version: log a predefined message
      Log.ignoreOutOfOrg();
   }
}
```

$$P \vdash \mathrm{logOutIf}(\varphi, \lambda) = \begin{cases} P \vdash \log(\lambda) & \text{if } P_{org} \vdash \mathrm{eval}(\varphi) = \text{true} \\ P \vdash \mathrm{nop} & \text{otherwise} \end{cases}$$

Preservation Condition

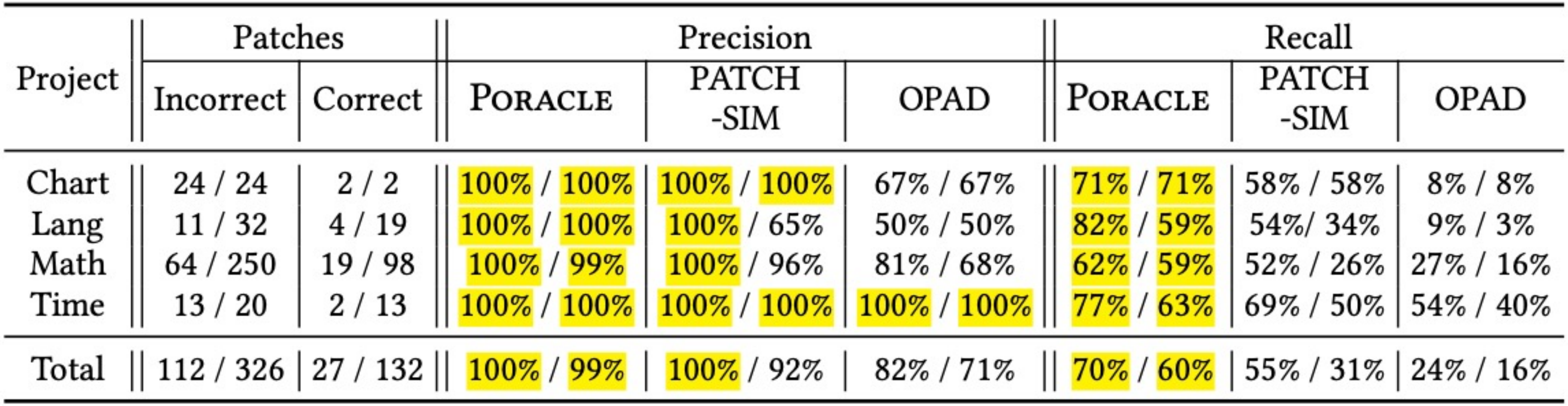


```
public void testGcd(int i, int j) {
    /* Original body:
    try {
        MathUtils.gcd(Integer.MIN_VALUE, 0);
        fail("expecting ArithmeticException");
    } catch (ArithmeticException expected) { // expected } */
    try {
        final long actual = MathUtils.gcd(i, j);
        boolean complement = !( (i==Integer.MIN_VALUE && j==0)
                || (i==0 && j==Integer.MIN_VALUE) );
        Log.logOutIf(complement, () –> new Long[] { actual });
    } catch (ArithmeticException e) {
        Log.logOutIf(!complement, () –> new String[] { e.toString() });
    } catch (Exception e) { Log.ignoreOutOfOrg(); }
}
```

실험 대상

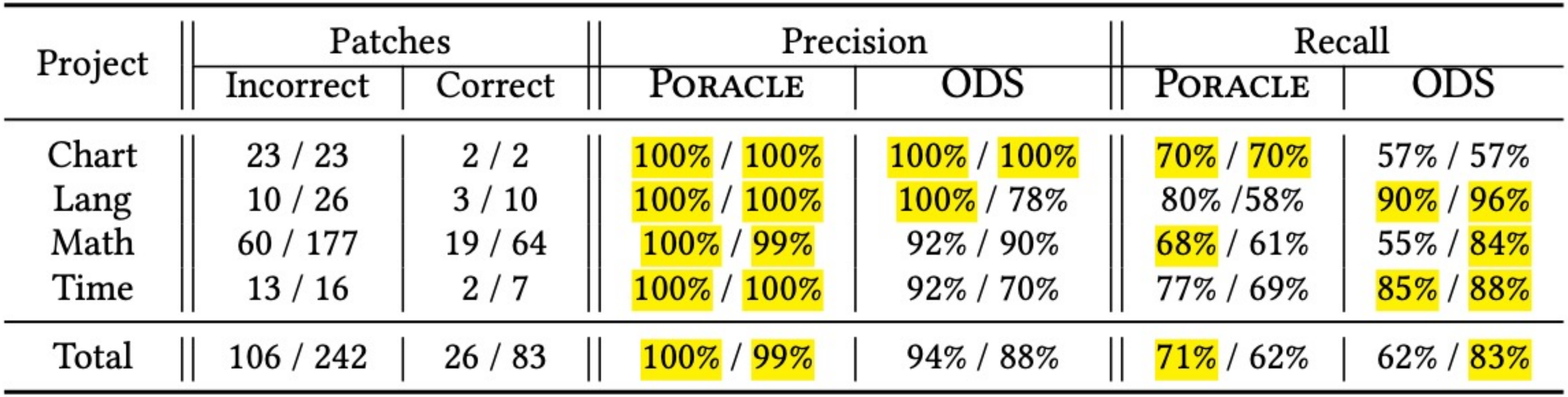
- PATCH-SIM[ICSE’18] 데이터셋
    - 139 patches (77 buggy versions)
    - 기존 연구[ICSE’20]에서 350 patches 추가

실험 결과



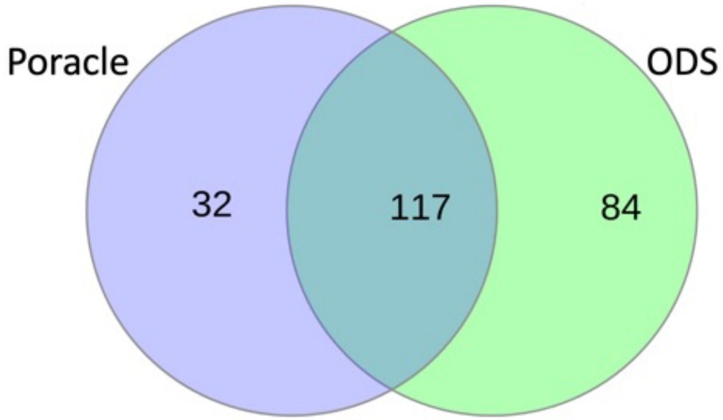
Project	Patches		Precision			Recall		
	Incorrect	Correct	Poracle	PATCH-SIM	OPAD	Poracle	PATCH-SIM	OPAD
Chart	24 / 24	2 / 2	100% / 100%	100% / 100%	67% / 67%	71% / 71%	58% / 58%	8% / 8%
Lang	11 / 32	4 / 19	100% / 100%	100% / 65%	50% / 50%	82% / 59%	54%/ 34%	9% / 3%
Math	64 / 250	19 / 98	100% / 99%	100% / 96%	81% / 68%	62% / 59%	52% / 26%	27% / 16%
Time	13 / 20	2 / 13	100% / 100%	100% / 100%	100% / 100%	77% / 63%	69% / 50%	54% / 40%
Total	112 / 326	27 / 132	100% / 99%	100% / 92%	82% / 71%	70% / 60%	55% / 31%	24% / 16%

실험 결과

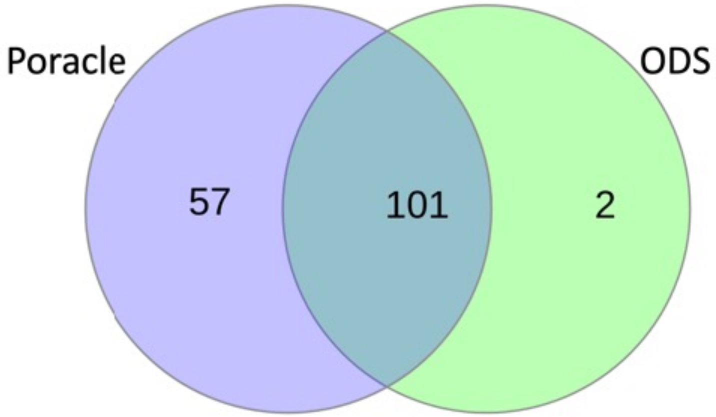


Project	Patches		Precision		Recall	
	Incorrect	Correct	PORACLE	ODS	PORACLE	ODS
Chart	23 / 23	2 / 2	100% / 100%	100% / 100%	70% / 70%	57% / 57%
Lang	10 / 26	3 / 10	100% / 100%	100% / 78%	80% /58%	90% / 96%
Math	60 / 177	19 / 64	100% / 99%	92% / 90%	68% / 61%	55% / 84%
Time	13 / 16	2 / 7	100% / 100%	92% / 70%	77% / 69%	85% / 88%
Total	106 / 242	26 / 83	100% / 99%	94% / 88%	71% / 62%	62% / 83%

실험 결과

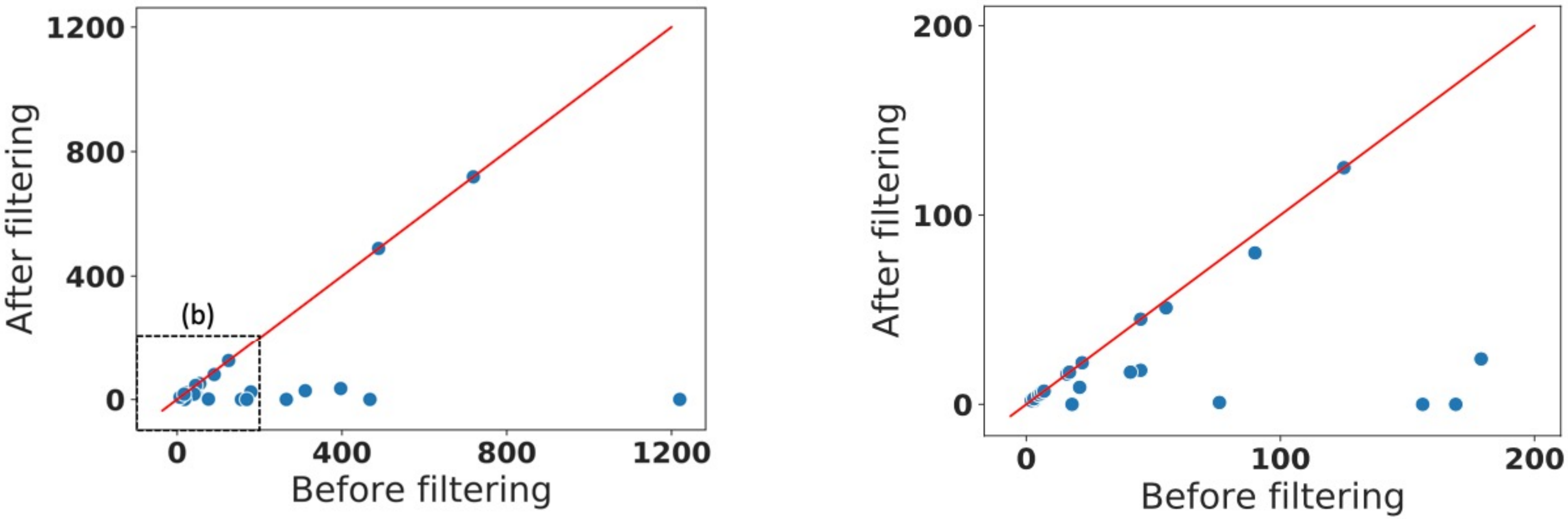


(a) Rightly rejected patches

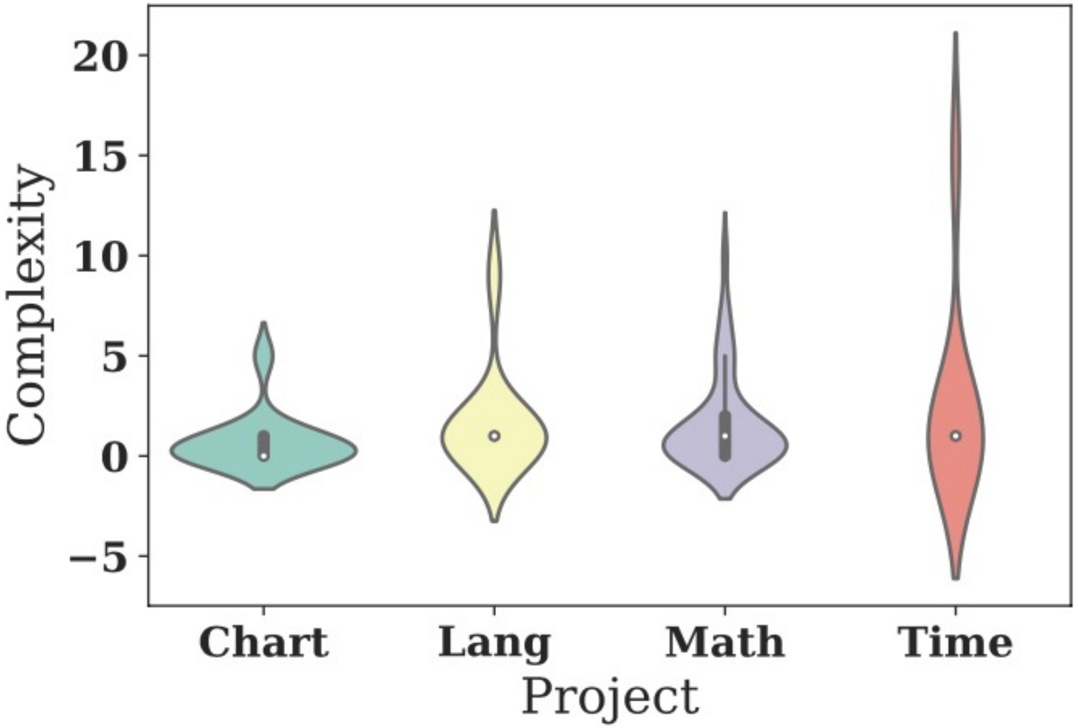


(b) Rightly accepted patches

JAID 실험 결과


1200
800
400
0
After filtering
(b)
0
400
800
1200
Before filtering
200
100
0
After filtering
0
100
200
Before filtering

Preservation Condition의 complexity



정리

- FAngelix: 패치 탐색 효율성 향상
- Verifix: 패치 정확성 보증
- Poracle: 패치 정확성 향상