Semantic Program Repair Using a Reference Implementation

Sergey Mechtaev[1] Manh-Dung Nguyen[2]
Yannic Noller[3] Lars Grunske[3]
Abhik Roychoudhury[1]

[1]National University of Singapore
{mechtaev,abhik}@comp.nus.edu.sg

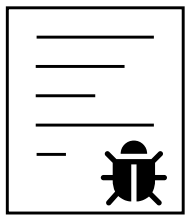
[2]Université Grenoble Alpes
nguyenmanhdung1710@gmail.com

[3]Humboldt University of Berlin
{yannic.noller,grunske}@informatik.hu-berlin.de

May 30, 2018

Methodology of test-driven program repair

Buggy program



Tests

Candidate patches

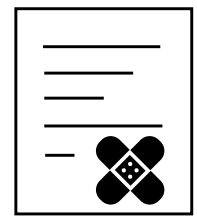
Search algorithm

Transformation schemas:

$\texttt{x} := \texttt{y} + \texttt{1}; \rightarrow \texttt{x} := \texttt{y} - \texttt{1};$

$\mathit{stmt}; \rightarrow \texttt{if}\ (x > 0)\ \mathit{stmt};$

...



Plausible patch
(passes all tests)

Test overfitting

Is the Cure Worse Than the Disease? Overfitting in Automated Program Repair. Smith et al. FSE'15

Assume an only test passes if the condition is false. How to correct the condition?

```
if (x > 5) {
   ...
}
```

```
if (false) {
   ...
}
```

```
if (x > 10) {
   ...
}
```

```
if (x > 5 && y > 5) {
   ...
}
```

Since a test suite is an incomplete specification, generated patches may overfit the tests.

Constraint solving and program synthesis

SAT

Satisfiability/satisfying assignment of boolean formulas:

$$(A_1 \vee \neg A_2) \wedge (A_2 \vee A_3 \vee A_4)$$

SMT

Satisfiability/satisfying assignment of first-order formulas:

$$((x = y) \wedge (y - z < 4)) \rightarrow (x - z < 6)$$

Program synthesis

Satisfiability/satisfying assignment of second-order formulas:

$$\exists p \in P. \bigwedge_{(\sigma, o) \in T} [\![p]\!]_\sigma = o$$

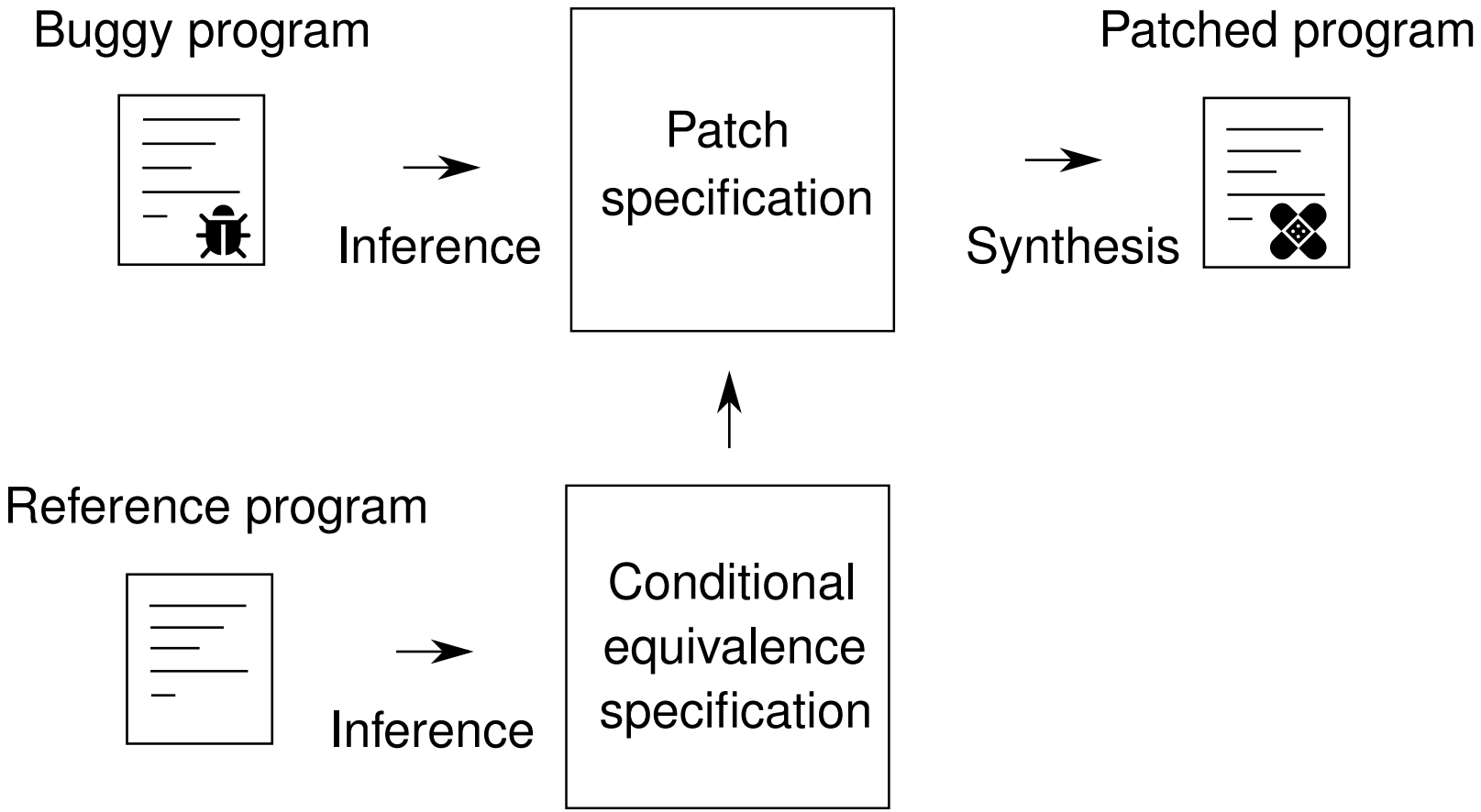
where P — a set of well-formed programs, T — a set of tests.

Semantic program repair

SemFix: Program Repair via Semantic Analysis. Nguyen et al. ICSE'13

Infer specification from reference implementation

Semantic Program Repair Using a Reference Implementation. Mechtaev et al. ICSE'18



Example: two search implementation

```
int lin_search(int x, int a[], int length) {
  int i;
  for (i=0; i<length; i++) {
    if (x == a[i])
      return i;
  }
  return -1;
}
```

```
int bin_search(int x, int a[], int length) {
  int L = 0;
  int R = length-1;
  do {
    int m = (L+R)/2;
    if (x == a[m]) {
      return m;
    } else if (x < a[m]) { // bug fix: x>a[m]
      L = m+1;
    } else {
      R = m-1;
    }
  } while (L <= R);
  return -1;
}
```

Repair problem formulation

Failing test

```
lin_search(2, [2, 4, 6], 3); ⇒ 0
bin_search(2, [2, 4, 6], 3); ⇒ -1
```

Repair enforcing conditional equivalence

Find `bin_search'` such that

$$\texttt{bin\_search}'(\gamma, [\alpha_0, \alpha_1, \alpha_2], \delta)$$

is equivalent to

$$\texttt{lin\_search}(\gamma, [\alpha_0, \alpha_1, \alpha_2], \delta)$$

for all inputs

$$\alpha_0 < \alpha_1 < \alpha_2 \wedge \delta = 3$$

Intrumenting buggy program

```
int lin_search(int x, int a[], int length) {
  int i;
  for (i=0; i<length; i++) {
    if (x == a[i])
      return i;
  }
  return -1;
}
```

```
int bin_search_instr(int x, int a[], int length) {
  int L = 0;
  int R = length-1;
  do {
    int m = (L+R)/2;
    if (x == a[m]) {
      return m;
    } else if (choose()) { // returns fresh variable β
      L = m+1;
    } else {
      R = m-1;
    }
  } while (L <= R);
  return -1;
}
```

Specification inference

Execute symbolically with precondition:

```
assume(α0 < α1 < α2 ∧ δ = 3);
lin_search(γ, [α0, α1, α2], δ);
```

Specification from reference program:

ID	$\pi^{\mathbf{r}}$	$\theta^{\mathbf{r}}_{out}$
...		
r_2	$\gamma \neq \alpha_0 \wedge \gamma = \alpha_1$	1
...		

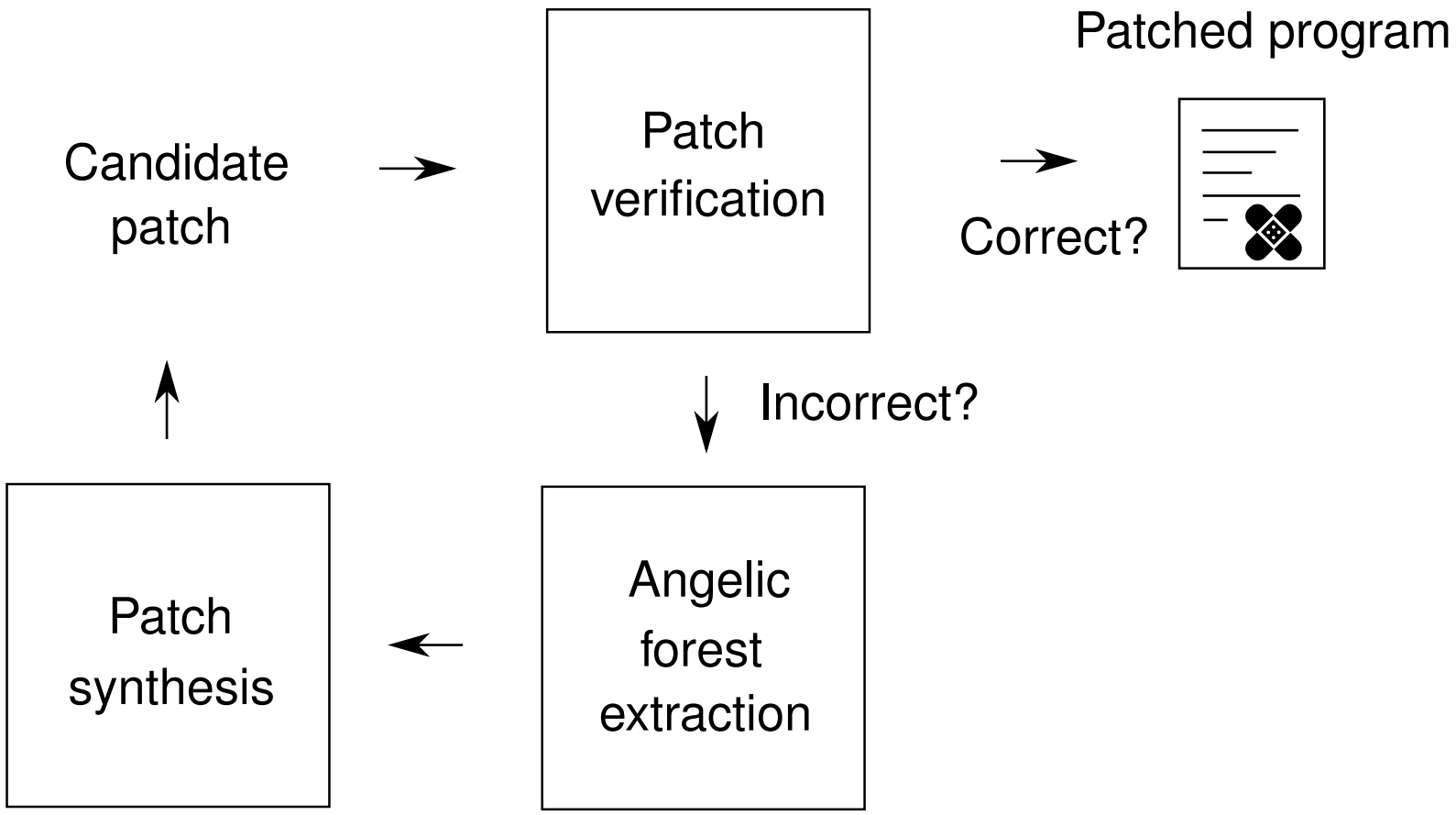
Execute symbolically with precondition:

```
assume(α0 < α1 < α2 ∧ δ = 3);
bin_search_intr(γ, [α0, α1, α2], δ);
```

Specification from buggy program:

ID	$\pi^{\mathbf{b}}$	$\theta_{\mathbf{c}}$	$\theta^{\mathbf{b}}_{out}$
...			
b_4	$\gamma \neq \alpha_1 \wedge \beta^0 \wedge \gamma \neq \alpha_2 \wedge \neg\beta^1$	$\beta^0 : \{x \mapsto \gamma, a[m] \mapsto \alpha_1\}$ $\beta^1 : \{x \mapsto \gamma, a[m] \mapsto \alpha_2\}$	-1
...			

Counterexample-guided inductive repair



Patch verification

$\texttt{x < a[m]}$ — candidate patch for `bin_search`

π^b — path condition of buggy program

θ^b_{out} — output of buggy program

π^r — path condition of reference program

θ^r_{out} — output of reference program

Then, verification condition for `x < a[m]` is

$$\forall\alpha_0\forall\alpha_1\forall\alpha_2\forall\gamma \bigwedge_{(\pi^r,\theta^r_{out})} \bigwedge_{(\pi^b,\theta^b_{out})} \pi^r \wedge \pi^b \wedge (\beta = [\![\texttt{x < a[m]}]\!]_{\theta_c}) \Rightarrow \theta^r_{out} = \theta^b_{out}$$

Counterexample:

$$\texttt{lin\_search}(-1,[-1,0,1],3) \neq \texttt{bin\_search}'(-1,[-1,0,1],3)$$

Patch synthesis

Baseline approach

To enable equivalence on input $(-1, [-1, 0, 1], 3)$, solve second-order formula (does not scale):

$$\exists e \in E. \alpha_0 = -1 \wedge \alpha_1 = 0 \wedge \alpha_2 = 1 \wedge \gamma = -1$$
$$\bigwedge_{(\pi^r, \theta^r_{out})} \bigwedge_{(\pi^b, \theta^b_{out})} \pi^r \wedge \pi^b \wedge (\beta = [\![e]\!]_{\theta_c}) \Rightarrow \theta^r_{out} = \theta^b_{out}$$

Optimized approach (Angelix, ICSE'16)

Extract angelic forest:

$$\{\{(\beta^i, c, \sigma)\}_\pi\}_t \quad \text{e.g. } \{(\beta^0, \mathit{False}, \{x \mapsto -1, a[m] \mapsto 0\})\}$$

Solve simpler second-order formula:

$$\exists e \in E. \bigwedge_t \bigvee_\pi \bigwedge_{(e^i, c, \sigma) \in \pi} [\![e]\!]_\sigma = c$$

Experiments with GNU Coreutils/Busybox

Busybox subject programs

Buggy program	Buggy commit	Angelix	SemGraft
sed	c35545a	Correct	Correct
seq	f7d1c59	Correct	Correct
sed	7666fa1	Incorrect	Correct
sort	d1ed3e6	Incorrect	Correct
seq	d86d20b	Incorrect	Correct
sed	3a9365e	Incorrect	Correct

Coreutils subject programs

Buggy program	Buggy commit	Angelix	SemGraft
mkdir	f7d1c59	Incorrect	Correct
mkfifo	cdb1682	Incorrect	Correct
mknod	cdb1682	Incorrect	Correct
copy	f3653f0	Correct	Correct
md5sum	739cf4e	Correct	Correct
cut	6f374d7	Incorrect	Correct

Example: bug in GNU Coreutils cut

GNU Coreutils wrongly interprets the command `-b 2-,3-` as `-b 3-` (extract input bytes staring from the third byte):

```
$ echo -ne '1234' | cut -b 2-,3-
34
```

instead of `-b 2-` (extract input bytes staring from the second byte):

```
$ echo -ne '1234' | cut -b 2-,3-
234
```

Developer tests:

```
echo -ne '1234' | cut -b 2-,3-
echo -ne '1234' | cut -b 3-,2-
```

Example: bug in GNU Coreutils cut

Automatic patch based on developer tests

```
if (!rhs_specified)
  {
    if (eol_range_start == 0 || eol_range_start == 3)
      eol_range_start = initial;
    field_found = true;
  }
```

Developer patch

```
if (!rhs_specified)
  {
    if (eol_range_start == 0 || initial < eol_range_start)
      eol_range_start = initial;
    field_found = true;
  }
```

Example: bug in GNU Coreutils cut

Introducing parameters into the test:

`echo -ne '1234' | cut -b` α_0`-,`α_1`-`

patch 1	$\texttt{eol\_range\_start} == 3$
cex 1	$-\texttt{b}\ 3-, 4-$
af 1	$(\beta^0, T, \{\texttt{initial} \mapsto 3, \texttt{eol\_range\_start} \mapsto 0, \}),$ $(\beta^1, F, \{\texttt{initial} \mapsto 4, \texttt{eol\_range\_start} \mapsto 3, \}).$
...	
patch N	$\texttt{initial} < \texttt{eol\_range\_start}$
verified	

$\texttt{initial} < \texttt{eol\_range\_start}$ enables equivalence of Coreutils cut and Busybox cut for all values of α_0 and α_1.

Availability of reference implementation

- Education
  - reference solution for student assignments
- Commodity software
  - standard library implementations
  - audio codecs
  - compression algorithms
  - parsers
  - network protocols
  - digital signal processing algorithms
  - web servers
  - database management systems
  - ...
- Backporting
  - patching forks of the same project

Program repair using reference implementation

reliable Provides additional guarantees compared with test-only approaches.

scalable User-defined input condition and angelic forest extraction.

Links

Automated program repair bibliography, tools, benchmarks:

`http://program-repair.org`

Angelix — program repair tool based on symbolic execution:

`http://angelix.io`