DirectFix: Looking for Simple Program Repairs

Sergey Mechtaev, Jooyong Yi, Abhik Roychoudhury

March 7, 2017

presenter name(s) removed for FERPA considerations

What is DirectFix?

- Automatic software repair program
  - GenProg
  - SemFix
  - PAR
- Why automate software repair?
  - Mozilla, 300 bugs daily

Similarity to existing?

- Test-driven method
- Automated repair

Difference from existing?

- Simplicity thus safer

Key Ideas

- For existing test-driven repair methods except DirectFix,
  - Statistical fault localization.
  - generate possible patches accordingly.

- For DirectFix,
  - fault localization + repair generation in an efficient way.
  - i.e. without enumerating all possible repairs.
  - insight of potential buggy functions.

A Buggy Program

```
1   x = E1; // E1 represents an expression.
2   y = E2; // E2 represents an expression.
3   S1; // S1 represents a statement. Neither x nor y is redefined by S1.
4   if (x > y) // FAULT: the conditional should be x >= y
5     return 0;
6   else
7     return 1;
```

(a) A buggy program snippet; a bug is in line 4.

Patch Generated by Genprog

```
1   x = E1; y = E2;
2   if (x == y) { S1; return 0; }  // This line is one possible repair.
3   S1;
4    if  (x > y)
5      return  0;
6   else
7      return  1;
```

(b) A repair that resembles a GenProg repair

Patch Generated by DirectFix

```
1   x = E1;
2   y = E2;
3   S1;
4   if (x >= y) // SIMPLE FIX: >= is substituted for >
5     return 0;
6   else
7     return 1;
```

(c) An alternative simpler repair; an operator is replaced.

Another Buggy Program

```
1    if (x > y)    // FAULT 1: the conditional should be x >= z
2      if (x > z)  // FAULT 2: the conditional should be x >= y
3        out = 10;
4      else
5        out = 20;
6    else out = 30;
7    return out;
```

(a) A buggy program snippet; bugs are in line 1 and 2.

Patch Generated by SemFix

```
1   if (x > y)
2     if (x > z)
3       out = 10;
4     else
5       out = 20;
6   else out = 30;
7   return ((x>=z)? ((x>=y)? 10 : 20) : 30); // This line is one possible repair.
```

(b) A repair that resembles a SemFix repair

Patch Generated by DirectFix

```
1    if (x >= z)    // SIMPLE FIX: >=z is substituted for >y
2      if (x >= y) // SIMPLE FIX: >=y is substituted for >z
3        out = 10;
4      else
5        out = 20;
6    else out = 30;
7    return out;
```

(c) An alternative simpler repair; operators and variables are replaced.

Third Buggy Program

```
1    // FAULT: k is NOT equal to the length of array s.
2    for (i=0; i<k; i++)
3      if (s[i] == c) return TRUE;
4    return FALSE;
```

(a) A buggy program that checks if the character c is included in string s.

Expected and Actual output

Input			Output	
s	c	k	expected	actual
"ab?"	'?'	2	TRUE	FALSE
"ab?c"	'?'	3	TRUE	TRUE
"!ab"	'!'	2	TRUE	TRUE

(b) Expected input and output

A repair that pass the above tests

```
1   for (i=0; i<k; i++)
2     // The following line is one possible repair.
3     if (c == '?' || c == '!') return TRUE;
4   return FALSE;
```

(c) A (buggy) repair that passes the above tests

A simplest repair

```
1   for (i=0; i<=k; i++) // SIMPLE FIX: <= is substituted for <
2     if (s[i] == c) return TRUE;
3   return FALSE;
```

(d) A more reliable repair

Statement Proposal (Hypothesis)

Simple repairs are less likely to change the correct behavior of the original version than more complex repairs. Thus, simple repairs are likely to be less hazardous

Solution Statement Proposal

- Iteratively generate a repair at each combination of suspicious program locations.
- Select simplest repair.

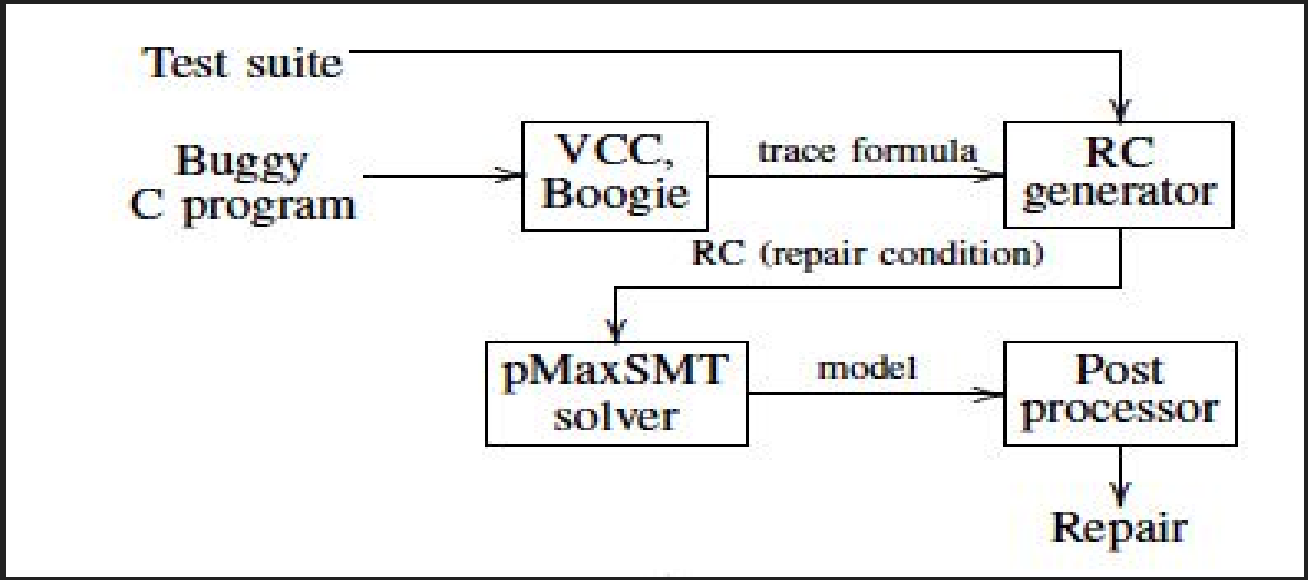
- Scalability?

Solution Statement Proposal

- Integrate the two phases of program repair
  - Fault localization
  - Repair search

- How to do these?

DirectFix WorkFlow



Experiment Result

TABLE II
EXPERIMENTAL RESULTS

Subject	Repairs			
	Total	Equivalent (E)	Same Loc (S)	Diff (D)
Tcas	36 (87%)	19 (54%)	33 (91%)	2.28
Replace	11 (37%)	9 (81%)	10 (91%)	2.54
Schedule	4 (44%)	4 (100%)	4 (100%)	2.5
Schedule2	2 (22%)	1 (50%)	2 (100%)	2
Coreutils	5 (56%)	0 (0%)	3 (60%)	2
Overall	59%	56%	89%	2.26

Result Comparison

TABLE III
COMPARISON WITH SEMFIX; E STANDS FOR EQUIVALENT, S STANDS FOR SAME LOC, D STANDS FOR DIFF, AND R STANDS FOR REGRESSION

Subject	Total	DirectFix				SemFix			
		E	S	D	R	E	S	D	R
Tcas	30	16	29	2.26	12	3	11	4.1	17
Replace	5	5	5	2.8	0	3	4	10.2	2
Schedule	4	2	4	2.5	1	1	4	8.5	3
Schedule2	2	1	2	2	1	1	2	5	2
Coreutils	4	0	3	2	-	0	0	4	-
Overall	44	53%	95%	2.31	31%	17%	46%	6.36	54%

Disadvantages

- Slower
  - DirectFix, 3 minutes 20 seconds
  - SemFix, 9 seconds
- More sizeable changes
  - Time would exhausted.
- Incorrect function designated to be suspicious function
  - Fail to find the repairs

Improvement?

- Search based method, Genprog
  - Aggressively narrow down the search space
- Semantic analysis based method, DirectFix
  - Find the smallest patch

To Improve Scalability

Discussion Question?

What about apply DirectFix on non-buggy program?

Discussion Question?

How DirectFix ensure its repaired output is relevant to original buggy version?

Discussion Question?

The runtime of the DirectFix compared to SemFix, which one is faster?

Discussion Question?

What about DirectFix on buggy program that require more sizable changes?

Discussion Question?

How to eliminate DirectFix constraint on more sizeable changes?

Reference

DirectFix: Looking for Simple Program Repairs
https://www.comp.nus.edu.sg/~abhik/pdf/ICSE15-directfix.pdf

Thank You