Staged Program Repair with Condition Synthesis

Fan Long and Martin Rinard
MIT EECS & CSAIL, USA
{fanl, rinard}@csail.mit.edu

ABSTRACT
We present SPR, a new program repair system that combines staged program repair and condition synthesis. These techniques enable SPR to work productively with a set of parameterized transformation schemas to generate and efficiently search a rich space of program repairs. Together these techniques enable SPR to generate correct repairs for over five times as many defects as previous systems evaluated on the same benchmark set.

1.1 Staged Program Repair (SPR)
We present SPR, a new program repair system that uses a novel staged program repair strategy to efficiently search a rich search space of candidate repairs. Three key techniques work synergistically together to enable SPR to generate successful repairs for a range of software defects. Together, these techniques enable SPR to generate correct repairs for over five times as many defects as previous systems evaluated on the same benchmark set.
Program repair is time consuming!
Program repair is time consuming!

But maybe we can repair programs automatically.
Research Questions

- Can automatic program repair tools be designed in a way that scales to
  - Real programs?
  - A variety of classes of real defects?
- Are there effective techniques for limiting the search space of program repair to increase scalability? Is staged program repair such a technique?
- Can techniques effectively prioritize certain plausible repairs to reduce the amount of time required to find program repairs?
Staged Program Repair
Key Ideas for Staged Program Repair

- Uses a set of predetermined transformation schemas to generate a search space that contains a large number of useful repairs.
- Integrates transformation schemas with techniques for condition synthesis to produce branching conditions for proposed program repairs.
- Updates proposed repair schemas with synthesized conditions to yield the final repaired program.
Example: An Absolute Value Function

```c
int absolute_value(int x) {
    return -x;
    return x;
}
```

Positive Tests:
```
absolute_value(-3) == 3
absolute_value(0) == 0
absolute_value(-10) == 10
```

Negative Tests:
```
absolute_value(5) == 5
```

Above is an obviously bugged implementation of absolute value in C.

To the right is a set of positive and negative test cases for this program.
Applying a Transformation Schema

```plaintext
int absolute_value(int x) {
    return -x;
    return x;
}
```

The transformation schema used here is called M-Guard.
Using Condition Synthesis

After condition synthesis:

```c
int absolute_value(int x) {
    if (1 && ! (x > 0)) {
        return -x;
    }
    return x;
}
```

As written by a developer:

```c
int absolute_value(int x) {
    if (x <= 0) {
        return -x;
    }
    return x;
}
```
Would this have worked in GenProg?

- Maybe!
- Remember, GenProg leverages template solutions in existing code
- Without other code, no!

Program repaired with SPR:

```c
int absolute_value(int x) {
    if (1 && !(x > 0)) {
        return -x;
    }
    return x;
}
```
Contributions

● A new method of resolving defects in a program by proposing a set of repairs that are most-likely to work.

● A set of transformation schemas that generate a search space with many useful repairs and integrate well with condition synthesis techniques.

● An algorithm for performing condition synthesis to efficiently search the space of possible conditions for use in program repairs.

● Experimental results validating the effectiveness of the contributed techniques for automatic program repair.
Evaluation
Evaluating Staged Program Repair

- 69 defects and 36 functionality changes (same benchmarks as GenProg)
- SPR is used for each defect/change
  - 12 hour time limit
  - Tested with and without a source code file

<table>
<thead>
<tr>
<th>App</th>
<th>LoC</th>
<th>Tests</th>
<th>Defects/Changes</th>
<th>SPR</th>
<th>SPR WSF</th>
<th>Gen Prog</th>
<th>AE</th>
<th>SPR</th>
<th>SPR WSF</th>
<th>Gen Prog</th>
<th>AE</th>
</tr>
</thead>
<tbody>
<tr>
<td>libtiff</td>
<td>77k</td>
<td>78</td>
<td>8/16</td>
<td>5/0</td>
<td>5/0</td>
<td>3/0</td>
<td>5/0</td>
<td>1/0</td>
<td>1/0</td>
<td>0/0</td>
<td>0/0</td>
</tr>
<tr>
<td>lighttpd</td>
<td>62k</td>
<td>295</td>
<td>7/2</td>
<td>3/1</td>
<td>4/2</td>
<td>4/1</td>
<td>3/1</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
</tr>
<tr>
<td>php</td>
<td>1046k</td>
<td>8471</td>
<td>31/13</td>
<td>16/1</td>
<td>19/1</td>
<td>5/0</td>
<td>7/0</td>
<td>9/0</td>
<td>9/0</td>
<td>0/0</td>
<td>1/0</td>
</tr>
<tr>
<td>gmp</td>
<td>145k</td>
<td>146</td>
<td>2/0</td>
<td>2/0</td>
<td>2/0</td>
<td>1/0</td>
<td>1/0</td>
<td>1/0</td>
<td>1/0</td>
<td>0/0</td>
<td>0/0</td>
</tr>
<tr>
<td>gzip</td>
<td>491k</td>
<td>12</td>
<td>4/1</td>
<td>2/0</td>
<td>2/0</td>
<td>1/0</td>
<td>2/0</td>
<td>0/0</td>
<td>1/0</td>
<td>0/0</td>
<td>0/0</td>
</tr>
<tr>
<td>python</td>
<td>407k</td>
<td>35</td>
<td>9/2</td>
<td>5/1</td>
<td>3/1</td>
<td>0/1</td>
<td>2/1</td>
<td>0/0</td>
<td>0/1</td>
<td>0/1</td>
<td>0/1</td>
</tr>
<tr>
<td>wireshark</td>
<td>2814k</td>
<td>63</td>
<td>6/1</td>
<td>4/0</td>
<td>4/0</td>
<td>1/0</td>
<td>4/0</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
</tr>
<tr>
<td>fbc</td>
<td>97k</td>
<td>773</td>
<td>2/1</td>
<td>1/0</td>
<td>1/0</td>
<td>1/0</td>
<td>1/0</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
<td>0/0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>69/36</strong></td>
<td><strong>38/3</strong></td>
<td><strong>40/4</strong></td>
<td><strong>16/2</strong></td>
<td><strong>25/2</strong></td>
<td><strong>11/0</strong></td>
<td><strong>12/1</strong></td>
<td><strong>1/1</strong></td>
<td><strong>2/1</strong></td>
<td><strong>1/1</strong></td>
<td><strong>2/1</strong></td>
</tr>
</tbody>
</table>
correct repairs from *staged program repair*.
Comparison with Existing Work

- Directly compare generated repairs to existing tools.
  - GenProg
  - AE
  - PAR
- Evaluate performance increase from condition value search

<table>
<thead>
<tr>
<th>Defect/Change</th>
<th>Repair Type</th>
<th>Condition Value Search</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On</td>
<td>Off</td>
</tr>
<tr>
<td>php-307562-307561</td>
<td>Replace†</td>
<td>0/139</td>
</tr>
<tr>
<td>php-307846-307853</td>
<td>Add Init†</td>
<td>0/126</td>
</tr>
<tr>
<td>php-307914-307915</td>
<td>Replace Print††</td>
<td>0/188</td>
</tr>
<tr>
<td>php-308734-308761</td>
<td>Guarded Control††</td>
<td>6/257</td>
</tr>
<tr>
<td>php-309516-309535</td>
<td>Add Init†</td>
<td>0/133</td>
</tr>
<tr>
<td>php-309579-309580</td>
<td>Change Condition††</td>
<td>1/64</td>
</tr>
<tr>
<td>php-309892-309910</td>
<td>Delete</td>
<td>1/144</td>
</tr>
<tr>
<td>php-310991-310999</td>
<td>Change Condition†</td>
<td>4/101</td>
</tr>
<tr>
<td>php-311346-311348</td>
<td>Redirect Branch†</td>
<td>1/89</td>
</tr>
<tr>
<td>libtiff-ee2ce5-b5691a</td>
<td>Add Control††</td>
<td>3/294</td>
</tr>
<tr>
<td>gmp-13420-13421</td>
<td>Replace†</td>
<td>0/515</td>
</tr>
<tr>
<td>php-308262-308315</td>
<td>Add Guard††</td>
<td>N/A</td>
</tr>
<tr>
<td>php-309111-309159</td>
<td>Copy†</td>
<td>N/A</td>
</tr>
<tr>
<td>php-309688-309716</td>
<td>Change Condition††</td>
<td>N/A</td>
</tr>
<tr>
<td>php-310011-310050</td>
<td>Copy and Replace††</td>
<td>N/A</td>
</tr>
<tr>
<td>libtiff-d13be-ccadf</td>
<td>Change Condition†</td>
<td>N/A</td>
</tr>
<tr>
<td>libtiff-5b021-3dfb3</td>
<td>Replace†</td>
<td>N/A</td>
</tr>
<tr>
<td>gzip-a1d3d4-f17cbd</td>
<td>Copy and Replace††</td>
<td>N/A</td>
</tr>
<tr>
<td>python-69783-69784</td>
<td>Delete</td>
<td>N/A</td>
</tr>
<tr>
<td>fbc-5458-5459</td>
<td>Change Condition††</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Thank you!

Any Questions?
Discussion
Discussion

How does the author’s implementation of staged program repair prioritize between possible repairs to try?
Discussion

Can we integrate a better heuristic for this process?
Discussion

Could *staged program repair* be applied to other programs and tests and get similar results?
Discussion

Could staged program repair successfully employ different techniques for condition synthesis? What impact might this have?
Discussion

Is it a good idea to extend the search space for *staged program repair*?
Thanks again!

Staged Program Repair with Condition Synthesis

Fan Long and Martin Rinard
MIT EECS & CSAIL, USA
{fanl, rinard}@csail.mit.edu

ABSTRACT
We present SPR, a new program repair system that combines staged program repair and condition synthesis. These techniques enable SPR to work productively with a set of parameterized transformation schemas to generate and efficiently search a rich space of program repairs. Together these techniques enable SPR to generate correct repairs for over five times as many defects as previous systems evaluated on the same benchmark set.

1.1 Staged Program Repair (SPR)
We present SPR, a new program repair system that uses a novel staged program repair strategy to efficiently search a rich search space of candidate repairs. Three key techniques work synergistically together to enable SPR to generate successful repairs for a range of software defects. Together, these techniques enable SPR to generate correct repairs for over five times as many defects as previous systems evaluated on the same benchmark set.