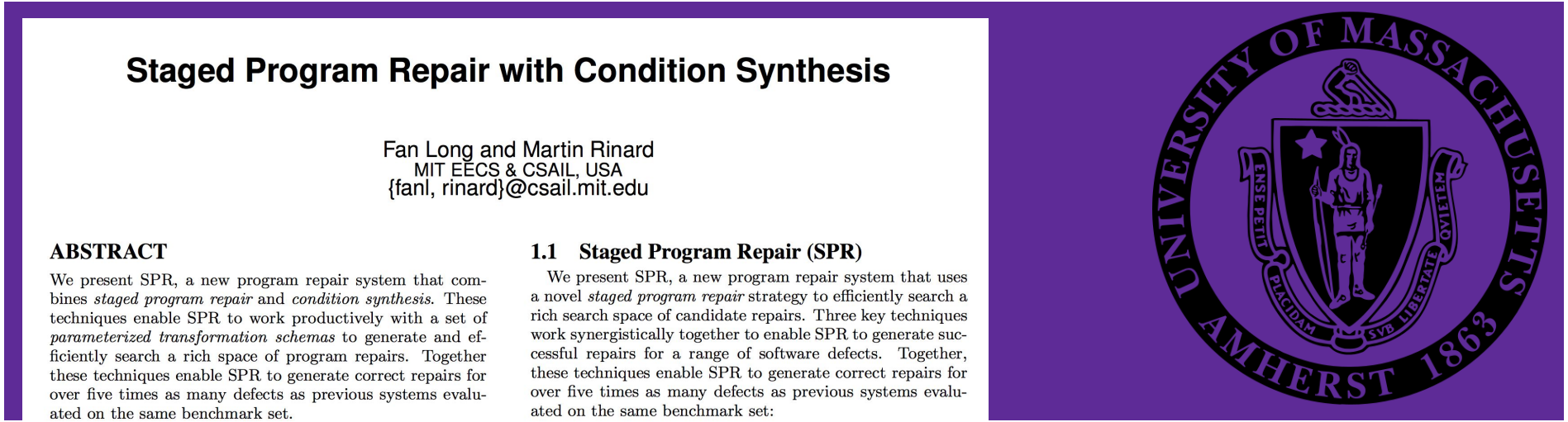
Staged Program Repair with Condition Synthesis

presenter name(s) removed for FERPA considerations



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

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

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.

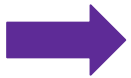
```
Positive Tests:
absolute_value(-3) == 3
absolute_value(0) == 0
absolute_value(-10) == 10



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

Applying a Transformation Schema

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



```
int absolute_value(int x) {
  if (1 && !abstc) {
    return -x;
  }
  return x;
}
```

The transformation schema used here is called M-Guard.

Using Condition Synthesis

After condition synthesis:

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

As written by a developer:

```
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:

```
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

App	LoC	Tests	Defects/ Changes	Plausible				Correct			
				SPR	SPR WSF	Gen Prog	AE	SPR	SPR WSF	Gen Prog	AE
libtiff	77k	78	8/16	5/0	5/0	3/0	5/0	1/0	1/0	0/0	0/0
lighttpd	62k	295	7/2	3/1	4/2	4/1	3/1	0/0	0/0	0/0	0/0
php	1046k	8471	31/13	16/1	19/1	5/0	7/0	9/0	9/0	1/0	2/0
gmp	145k	146	2/0	2/0	2/0	1/0	1/0	1/0	1/0	0/0	0/0
gzip	491k	12	4/1	2/0	2/0	1/0	2/0	0/0	1/0	0/0	0/0
python	407k	35	9/2	5/1	3/1	0/1	2/1	0/0	0/1	0/1	0/1
wireshark	2814k	63	6/1	4/0	4/0	1/0	4/0	0/0	0/0	0/0	0/0
fbc	97k	773	2/1	1/0	1/0	1/0	1/0	0/0	0/0	0/0	0/0
Total			69/36	38/3	40/4	16/2	25/2	11/0	12/1	1/1	2/1

19

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

Defect/ Change	Repair Type	Condition Value Search	
		On	Off
php-307562-307561	Replace†	0/139	6.3X
php-307846-307853	Add Init†	0/126	3.2X
php-307914-307915	Replace Print†‡	0/188	67.5X
php-308734-308761	Guarded Control†‡	6/257	4.5X
php-309516-309535	Add Init†	0/133	5.8X
php-309579-309580	Change Condition†‡	1/64	34.0X
php-309892-309910	Delete	1/144	25.5X
php-310991-310999	Change Condition†	4/101	68.9X
php-311346-311348	Redirect Branch†	1/89	42.4X
libtiff-ee2ce5-b5691a	Add Control†‡	3/294	94.1X
gmp-13420-13421	Replace†‡	0/515	3.7X
php-308262-308315	Add Guard†‡	N/A	6.9X
php-309111-309159	Copy‡	N/A	2.9X
php-309688-309716	Change Condition†‡	N/A	4.0X
php-310011-310050	Copy and Replace†‡	N/A	4.5X
libtiff-d13be-ccadf	Change Condition†	N/A	121.4X
libtiff-5b021-3dfb3	Replace†	N/A	5.5X
gzip-a1d3d4-f17cbd	Copy and Replace†‡	N/A	5.0X
python-69783-69784	Delete	N/A	40.6X
fbc-5458-5459	Change Condition†‡	N/A	18.8X

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: