# Towards Practical Program Repair with On-Demand Candidate Generation

**Jinru Hua, Mengshi Zhang, Kaiyuan Wang and Sarfraz Khurshid**

The University of Texas at Austin, USA

{lisahua | mengshi.zhang | kaiyuanw | khurshid}@utexas.edu

**Lucas Roque**

**2018**

I4Soft
Intelligence for Software

UFG

# Introdução

- Técnicas de reparo que modificam um programa defeituoso para corrigi-lo dado um conjunto de teste, pode reduzir substancialmente o custo de depuração manual;
- A abordagem mais comum, gera um conjunto de programas candidatos com possíveis correções e os valida nos casos de teste até encontrar o que passa em todos eles;
- É conceitualmente simples, mas com relativa baixa efetividade.

# Introdução

- Invés de recompilar e reexecutar cada candidato, SketchFix traduz o programa defeituoso para *sketches*;
- A idéia chave é que o espaço de soluções candidatas pode ser substancialmente podado utilizando informações do comportamento em tempo de execução e gerando candidatos *on-demand* durante a execução do teste;
- Exemplo: Consertar um erro que pode estar na condição ou no corpo de um *while*.

# Introdução

- Ao nível de nó na AST, SketchFix: Traduz um programa defeituoso em Java para *Sketches*, que serão completadas através de um sintetizador de acordo com o conjunto de teste.
- Dado um programa defeituoso e um conjunto de teste:
  - Introduz "*holes*" nos *statements* suspeitos;
  - Para preencher esses *"holes"*, e utilizado um sintetizador chamado EdSketch.
- Baseado na idéia de a correção é semanticamente próxima do programa original, são priorizados *schemas* que introduzem menos perturbação.

# Introdução

- Contribuições:
  - **Geração On-Demand de candidatos ao reparo:** Utilização de comportamentos em tempo de execução para geração de candidatos conforme necessário;
  - **Reparo a nível de nó na AST:** Consegue gerar reparos em uma granularidade “refinada”, resultando em patches de alta qualidade e semanticamente próximo ao original;
  - **Redução prática de programas sintetizados:** Transformação de amostras defeituosas em *sketches* e síntese e código para completá-lo *on-demand*.

# Exemplo

(A) Part of the human-written patch to fix the Chart14 defect

```
1.public class CategoryPlot extends Plot...{...
2. public boolean removeDomainMarker (...,boolean notify) {
3.  ArrayList markers;
4.  if (...) {...} else {
5.+ if (markers == null)
6.+  return false;
7. ...} }
```

(B) A sketch generated by SketchFix and synthesized solutions

```
1.public class CategoryPlot extends Plot...{...
2. public boolean removeDomainMarker (...,boolean notify) {
3.  ArrayList markers;
4.  if (...) {...} else {
5.  if(SketchFix.COND(ArrayList.class,new Object[]{markers,..}))
6.   return (Boolean) SketchFix.EXP(Boolean.class,
      new Object[]{markers,..};
7.  ...} }
  // Synthesized solution:
  // SketchFix.COND: markers==null,...
  // SketchFix.EXP: false,...
```

- Figura (A) apresenta uma correção humana de um bug que omite a checagem de um *null pointer* para um ArrayList;
- Caso uma ferramenta queira colocar um *if* e um retorno:
  - A classe possui 54 variáveis, gerando mais de 5 mil possibilidade de checagem;
  - 15 candidatos a retorno;
  - 15 dias para checar as 87 mil possibilidades.

# Exemplo

(A) Part of the human-written patch to fix the Chart14 defect

```
1.public class CategoryPlot extends Plot...{...
2. public boolean removeDomainMarker (...,boolean notify) {
3.  ArrayList markers;
4.  if (...) {...} else {
5.+ if (markers == null)
6.+  return false;
7. ...} }
```

(B) A sketch generated by SketchFix and synthesized solutions

```
1.public class CategoryPlot extends Plot...{...
2. public boolean removeDomainMarker (...,boolean notify) {
3.  ArrayList markers;
4.  if (...) {...} else {
5.  if(SketchFix.COND(ArrayList.class,new Object[]{markers,..}))
6.   return (Boolean) SketchFix.EXP(Boolean.class,
      new Object[]{markers,..};
7.  ...} }
  // Synthesized solution:
  // SketchFix.COND: markers==null,...
  // SketchFix.EXP: false,...
```

- Para explorar esse espaço de busca de forma mais eficiente, SketchFix transforma o programa em *sketches* com *holes* e os sintetiza *on-demand*;
- Cada sketch representa milhares de candidatos concretos;
- Na figura (B) apresenta 1 sketch representando 87 mil candidatos.
- Neste exemplo, Sketch Fix encontrou a primeira solução em 40 segundos, compilando 1 vez e executando os teste 2 vezes.

# **Proposta** | Transformação AST

- Sintaxe das expressões parciais (holes):
  - **Expression Holes:** Variáveis, constantes e dereferências;
    - SketchFix.EXP().
  - **Operator Holes:** Operadores aritmeticos, relacionais e lógicos.
    - SketchFix.AOP(), SketchFix.ROP(), SketchFix.BOP().

| | | |
|---|---|---|
| atomic expr | $e$ := | $var \mid const \mid var.f$ |
| constant | $const$ := | $null \mid true \mid false \mid k$ |
| arithmetic op | $aop$ := | $+ \mid - \mid \times \mid / \mid \%$ |
| relational op | $rop$ := | $== \mid != \mid > \mid < \mid \leqslant \mid \geqslant$ |
| logical op | $lop$ := | $\&\& \mid \mid\mid$ |
| composite expr | $e$ := | $e_1\ op\ e_2$ or $array[e_{int}]$ |

# **Proposta** | Transformação AST

- Esquemas de transformação:

$$M_{exp} = \frac{p[\ell] \vdash e_t}{e_t \mapsto \omega_t}$$

$$M_{op} = \frac{p[\ell] \vdash op}{op \mapsto \delta}$$

$$M_{par} = \frac{p[\ell] \vdash f(par),\ f(par) \vdash f'(par \cup e_t)}{f(par) \mapsto f'(par \cup \omega_t)}$$

$$M_{con} = \frac{p[\ell] \vdash \texttt{if}\ (c)}{c \mapsto c\ \texttt{lop}\ (\omega_t\ \texttt{rop}\ \omega'_t)}$$

$$M_{if} = \frac{p(\ell) \vdash (v,t)}{p(\ell) \mapsto \texttt{if}\ (\omega_t\ \texttt{rop}\ \omega'_t)\ p(\ell)}$$

$$M_{rtn} = \frac{p(\ell) \vdash (v,t)}{p(\ell) \mapsto \texttt{return}\ \omega_t\ p(\ell)}$$

- $M_{exp}$ - true → ??
  - SketchFix.EXP(t, new Object[]{v1,v2,..});
- $M_{op}$ - (a > b) → (a ?? b)
  - SketchFix.ROP(Integer.class, new Object[]{a,b});
- $M_{par}$ - Métodos com mais de uma assinatura;

# **Proposta** | Transformação AST

- Esquemas de transformação:

$$M_{exp} = \frac{p[\ell] \vdash e_t}{e_t \mapsto \omega_t}$$

$$M_{op} = \frac{p[\ell] \vdash op}{op \mapsto \delta}$$

$$M_{par} = \frac{p[\ell] \vdash f(par),\ f(par) \vdash f'(par \cup e_t)}{f(par) \mapsto f'(par \cup \omega_t)}$$

$$M_{con} = \frac{p[\ell] \vdash \texttt{if}\ (c)}{c \mapsto c\ \texttt{lop}\ (\omega_t\ \texttt{rop}\ \omega'_t)}$$

$$M_{if} = \frac{p(\ell) \vdash (v,t)}{p(\ell) \mapsto \texttt{if}\ (\omega_t\ \texttt{rop}\ \omega'_t)\ p(\ell)}$$

$$M_{rtn} = \frac{p(\ell) \vdash (v,t)}{p(\ell) \mapsto \texttt{return}\ \omega_t\ p(\ell)}$$

- $M_{con}$ - Nova cláusula para condição:
  - Primitivos: operações relacionais;
  - Lógicos: operações lógicas.
- $M_{if}$ - Introduz uma condição *if*:
  - SketchFix.COND(t, new Object[]{v1,v2,...});
- $M_{rtn}$ - Adiciona um retorno:
  - SketchFix.EXP(t, new Object[]{v1,v2,...});

# **Proposta** | Transformação AST

**Algorithm 1:** Static Transformation for Sketch Generation

```
Input  : Faulty program p, Fault locations L, Schemas M
Output : List of sketches Q
1  Function transformSketch (p, L, M) is
2      Q ← ∅;
3      foreach ℓ ∈ L do
           /* apply one schema                          */
4          foreach σ ∈ M do
5              Q ← Q ∪ σ(p, ℓ) ;
           /* apply two schemas                         */
6          i ← 0;
7          while i < M.size do
8              ω ← M[i](p, ℓ) ;
9              j ← i ;
10             while j < M.size do
11                 Q ← Q ∪ M[j](ω, ℓ), j++ ;
12             i++;
```

- Para lidar com defeitos que precisam de múltiplos “holes”, é aplicado transformações incrementais no *statement*;
- Nunca será aplicado mais de duas transformações por local;
- Serão priorizados *schemas* com menor perturbação.

# **Proposta** | Geração dos Candidatos

**Algorithm 2:** On-Demand Candidate Generation based on EdSketch [13]

**Input** : Sketches $P$, test suite $T$
**Output**: Complete Program $P'$ that pass all test cases

```
1  Function synthesizeHole (hole) is
2      if hole.candidates==null then
           /* First Access                                   */
3          hole.candidates ← candidateGen(hole);
4      if hole.id == -1 then
           /* First Access                                   */
5          hole.id ← choose(0, hole.candidates.size-1) ;
6      return hole.candidates[hole.id] ;
7  Function sketch () is
8      do
9          try
10             exploreCurrentChoice();
11         catch BacktrackException
12             createNextChoice() ;
13     while incrementCounter();
14 Function exploreCurrentChoice() is
15     try
16         foreach test ∈ T do
17             test.run() ;
18     catch TestFailureException
19         throw BacktrackException;
20     printSolution() ;
21     searchExit(); /* if only needs the first solution   */
```

- **Linha 2**: Quando a execução do teste encontra um *hole*, são gerados os candidatos baseados nas variáveis visíveis;
- **Linha 3**: Retorna um vetor de candidatos;
- **Linha 4**: Caso um candidatos não tenha sido selecionado, ele será de forma não determinística pela **linha 5**.

# **Proposta** | Geração dos Candidatos

**Algorithm 2:** On-Demand Candidate Generation based on EdSketch [13]

```
   Input  : Sketches P, test suite T
   Output : Complete Program P' that pass all test cases
1  Function synthesizeHole (hole) is
2      if hole.candidates==null then
           /* First Access                                   */
3          hole.candidates ← candidateGen(hole);
4      if hole.id == -1 then
           /* First Access                                   */
5          hole.id ← choose(0, hole.candidates.size-1) ;
6      return hole.candidates[hole.id] ;
7  Function sketch () is
8      do
9          try
10             exploreCurrentChoice();
11         catch BacktrackException
12             createNextChoice() ;
13     while incrementCounter();
14 Function exploreCurrentChoice() is
15     try
16         foreach test ∈ T do
17             test.run() ;
18     catch TestFailureException
19         throw BacktrackException;
20     printSolution() ;
21     searchExit(); /* if only needs the first solution   */
```

- **Linha 10**: Este candidato será explorado até que haja uma exceção, ou algum caso de teste falhe;
- **Linha 12**: Quando ocorrer uma BackTrackExpetion na **linha 19**, o processo será reiniciado;
- A processo acaba quando o espaço de soluções candidatas acaba, ou encontra-se um programa que satisfaça todos os casos de teste.

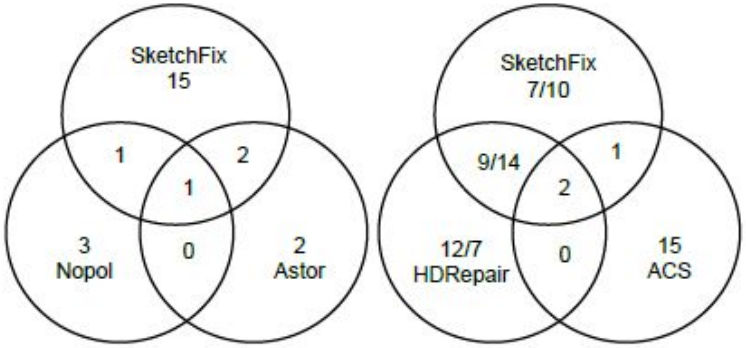
# Avaliação

- **RQ1:** Qual a eficácia do SketchFix comparada a outras técnicas?
- **RQ2:** A geração *on-demand* de candidatos reduz o espaço de busca?
- **RQ3:** Como a transformação de baixa granularidade afeta a qualidade dos reparos gerados?

# Avaliação

- A avaliação foi realizada utilizando o *benchmark* Defects4J, com 357 defeitos reais em 5 aplicações Java;
- Foi usado um *framework* de análise de *bytecode* com um agente Java para capturar a cobertura da execução do conjunto de teste;
- E a técnica de localização de defeitos Ochiai, que tem se mostrado mais efetiva em programas orientados a objeto.

# **RQ1** | Eficácia do Reparo




HDRepair [25] assumes that the faulty methods are known in advance. SketchFix fixes 8 more defects with this assumption, 5 of them are also fixed by HDRepair. We report two results without and with the assumption.

- SketchFix foi comparado com Astor, Nopol, ACS e HDRepair;

# **RQ1** | Eficácia do Reparo (Análise Manual)

| No. | SF | A | N | C | H |
|---|---|---|---|---|---|
| CH1 | ✓ | ? | × | × | ✓ |
| CH3 | × | ? | ? | × | × |
| CH5 | × | ? | × | × | × |
| CH7 | × | ? | × | × | × |
| CH8 | ✓ | × | × | × | ✓ |
| CH9 | ✓ | × | × | × | × |
| CH11 | ✓ | × | × | × | × |
| CH13 | ? | ? | ? | × | × |
| CH14 | × | × | × | ✓ | × |
| CH15 | × | ? | × | ✓ | × |
| CH19 | × | × | × | × | × |
| CH20 | ✓ | × | × | × | × |
| CH21 | × | × | ? | × | × |
| CH24 | ✓ | × | × | × | × |
| CH25 | × | ? | ? | × | × |
| CH26 | ?* | ? | ? | × | × |

| No. | SF | A | N | C | H |
|---|---|---|---|---|---|
| L6 | ✓ | × | × | × | ✓ |
| L7 | × | × | ? | ✓ | × |
| L10 | ×* | × | × | × | ✓ |
| L24 | × | × | × | ✓ | × |
| L35 | × | × | × | ✓ | × |
| L39 | × | × | × | × | × |
| L43 | × | × | × | × | ✓ |
| L44 | × | × | ✓ | × | × |
| L46 | × | × | ? | × | × |
| L51 | ?* | × | ? | × | ✓ |
| L53 | × | × | ? | × | × |
| L55 | ✓ | ? | ✓ | × | × |
| L57 | × | × | × | × | ✓ |
| L58 | × | × | ✓ | × | × |
| L59 | ✓ | × | × | × | ✓ |

| No. | SF | A | N | C | H |
|---|---|---|---|---|---|
| T4 | ? | ? | × | × | × |
| T11 | × | ? | ? | × | × |
| T15 | × | × | × | ✓ | × |
| T19 | ×* | × | × | × | ✓ |

# **RQ2** | Redução do Espaço de Busca

| # | | Fix | Type | Sk. | Sp. | cSk. | Exe | Cor | T(m) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | CH1 | C | $M_{rop}$ | 1.7k | 301.2k | 209 | 2.8k | 2.9k | 208.5 |
| 2 | CH8 | C | $M_{exp}$ | 607 | 8.7k | 83 | 663 | 663 | 32.9 |
| 3 | CH9 | C | $M_{con}$ | 744 | 6.9k | 678 | 100 | 100 | 5.6 |
| 4 | CH11 | C | $M_{exp}$ | 249 | 2.4k | 25 | 26 | 31 | 7.2 |
| 5 | CH13 | P | $M_{par}$ | 813 | 79.8k | 115 | 914 | - | - |
| 6 | CH20 | C | $M_{exp}$ | 137 | 2.9k | 127 | 206 | 206 | 14.9 |
| 7 | CH24 | C | $M_{exp}$ | 17 | 378 | 1 | 4 | 4 | 0.5 |
| 8 | CH26 | P | $M_{if}$, $M_{rtn}$ | 77 | 1.6k | 21 | 105 | - | - |
| 9 | C14 | C | $M_{exp}$ | 2.6k | 46.1k | 144 | 4 | 36 | 40.7 |
| 10 | C62 | C | $M_{rop}$ | 490 | 4.9k | 58 | 72 | 92 | 8.8 |
| 11 | C70 | P | $M_{par}$ | 1.2k | 18.9k | 108 | 28 | - | - |
| 12 | C73 | P | $M_{rop}$ | 476 | 5.2k | 59 | 40 | - | - |
| 13 | C126 | C | $M_{con}$ | 462 | 5.6k | 32 | 8 | 64 | 7.3 |
| 14 | L6 | C | $M_{exp}$ | 255 | 3.0k | 1 | 51 | 424 | 2.9 |
| 15 | L51 | P | $M_{if}$, $M_{rtn}$ | 222 | 2.4k | 43 | 28 | - | - |
| 16 | L55 | C | $M_{con}$ | 108 | 1.6k | 76 | 62 | 71 | 86.2 |
| 17 | L59 | C | $M_{exp}$ | 188 | 1.8k | 48 | 5 | 5 | 0.1 |
| 18 | M5 | C | $M_{exp}$ | 1 | 72 | 1 | 1 | 1 | 0.1 |
| 19 | M33 | C | $M_{par}$ | 1.1k | 17.8k | 446 | 221 | 631 | 20.2 |
| 20 | M50 | C | $M_{con}$ | 655 | 6.0k | 43 | 22 | 92 | 0.9 |
| 21 | M59 | C | $M_{exp}$ | 259 | 4.0k | 36 | 7 | 7 | 0.4 |
| 22 | M70 | C | $M_{par}$ | 139 | 2.8k | 60 | 8 | 8 | 0.1 |
| 23 | M73 | P | $M_{if}$, $M_{rtn}$ | 383 | 6.2k | 76 | 59 | - | - |
| 24 | M82 | C | $M_{rop}$ | 500 | 6.0k | 68 | 36 | 803 | 12.1 |
| 25 | M85 | C | $M_{rop}$ | 407 | 5.2k | 78 | 590 | 590 | 23.1 |
| 26 | T4 | P | $M_{par}$ | 555 | 23.4k | 40 | 274 | - | - |

*Fix* represents whether it is a correct fix (*C*) or a plausible fix (*P*). *Type* denotes the schema types that yield the repair. *Sk.* shows the number of generated sketches. *Sp.* presents the total search space of candidates. *cSk.* is the number of compiled sketches when SketchFix generates first repair. *Exe* represents the number of candidates SketchFix explores when it generates the first repair passing all tests. *Cor* represents that number when SketchFix generates the first correct repair based on the manual inspection. *T(m)* reports the performance time to synthesize a correct fix.

# **RQ3** | Qualidade dos Reparos

(A) A human-written patch for the defect Math33

```
public class Precision {
 public static int compareTo(double x,double y,double eps)..
 public static int compareTo(double x,double y,int maxUlps)..
 }
 /* SimplexTableau.java */
 private final int maxUlps;
 private final double epsilon;
 protected void dropPhase1Objective() {
-  if (Precision.compareTo(entry, 0d, maxUlps)>0){...
+  if (Precision.compareTo(entry, 0d, epsilon)>0){...
}
```

(B) A sketch generated by SketchFix and a synthesized solution

```
 /* SimplexTableau.java as sketch */
protected void dropPhase1Objective() {
 if (Precision.compareTo(entry, 0d, (Double) SketchFix.EXP(
  Double.class, new Object[]{..,epsilon,maxUlps,..}))>0){..
 }
 // Synthesized solution: SketchFix.EXP: epsilon
```

(C) A plausible repair generated by Nopol

```
 protected void dropPhase1Objective() {
  if (Precision.compareTo(entry, 0d, maxUlps) > 0) {
+  if (numSlackVariables<constraints.size()) {...}}
```

- Astor, ACS e HDRepair não conseguiram reparar a falha;
- Nopol gera um reparo plausível.

# Conclusões

- O trabalho introduz uma técnica de reparo *on-demand*;
- Utiliza informações em tempo de execução para podar uma parte do espaço de busca;
- A técnica reduz reparo para síntese transformando programas defeituosos para sketches na granularidade no nível do nó na AST;
- Experimentos demonstram que SketchFix trabalha melhor com manipulação de expressões;
- E que a transformação de baixa granularidade produz *patches* de alta qualidade;

# Perguntas?

Obrigado!