Cause Points Analysis for Effective Handling of Alarms

Tukaram Muske, Uday P. Khedker

TRDDC, Tata Consultancy Services, India
Indian Institute of Technology Bombay, India

ISSRE 2016, Ottawa, Oct 23-27
Motivating Example

```c
const int arr1[]={0,3,7,9,14,22,34};
char arr2[35], str[20], bound, tmp;

void foo(){
    unsigned int i, j, k, length;
    ... // some code
    scanf("%s",str); //Cause point CP7
    if (i < 7 && j < i)
        bound=arr1[i]-arr1[j]; //Cause point CP9
    for(k = 0; k <= bound; k++) { //OFUF
        f1(k);
    }
    length = strlen(str);
    f2(bound, length);
}

void f1(int p){
    if(nondet()) arr2[p] = 0; //AIOB
    else arr2[p] = 1; //AIOB
}

void f2(int p, unsigned int q){
    arr2[p - 1] = 100 / q; //AIOB, ZD
    tmp = str[q]; //AIOB
}
```

6 alarms due to 2 cause points
\[ \varepsilon_{\text{orig}} = ((\varepsilon_{O11} + E_{cp9}) + \varepsilon_{A19} + \varepsilon_{A20} + \varepsilon_{A24}) + ((\varepsilon_{Z24} + E_{cp7}) + \varepsilon_{A25}) \]
Proposed Approach

- Interactive static analysis
- Iterative static analysis
- Cause point Analysis
Motivating Example

```c
1  const int arr1[]={0,3,7,9,14,22,34};
2  char arr2[35], str[20], bound, tmp;
3  
4  void foo(){
5  unsigned int i, j, k, length;
6  ... // some code
7  scanf("%s",str);  //Cause point CP7
8  if (i < 7 && j < i)
9      bound=arr1[i]-arr1[j]; //Cause point CP9
10  
11  for(k = 0; k <= bound; k++) {  //OFUF
12      f1(k);
13  }
14  length = strlen(str);
15  f2(bound, length);
16 }
17
18  void f1(int p){
19      if(nondet()) arr2[p] = 0;  //AIOB
20      else arr2[p] = 1;  //AIOB
21  }
22  
23  void f2(int p, unsigned int q){
24      arr2[p - 1] = 100 / q;  //AIOB, ZD
25      tmp = str[q];  //AIOB
26  }
```

Get values for unknowns from user

Re-analyze the code using inputs

No alarm is generated
Manual Effort Reduction

\[ \varepsilon_{\text{orig}} = \]

\[ \varepsilon_{\text{new}} = \]

\[ \varepsilon_{\text{saved}} = \]
**Cause Points-specific Queries**

1. **Cause point CP₉**
   - **Expression:** $E_{cp₉}$
   - **Query:** “What are the values computed by ‘$arr[i] - arr[j]$’ at line 9?”
   - **Values:** $arr2[p], arr2[p - 1]$ where Arraysize = 35

2. **Cause point CP₇**
   - **Expression:** $E_{cp₇}$
   - **Query:** “What are the values given to ‘str’ at line 7?”
   - **Values:** 100/q, str[q] where Arraysize = 20

3. **New Change:** $\varepsilon_{new}$

4. **Does the values computed by ‘$arr[i] - arr[j]$’ lie in the range of $\langle 1;34 \rangle$?**
   - **Response:**
   - **Is the input string ‘str’ always nonempty and contains less than 20 characters?**
   - **Response:**
Modeling of Cause Points

- **Unknowns of various types**
  - *i-unknowns*  
    - e.g. `scanf("%s", str);`
  - *c-unknowns*  
    - e.g. `ratio * a * b + size`
  - *loop-unknowns*  
    - e.g. `for(i=0; i < 10 || foo(); i++){…}`
  - *ds-unknowns*  
    - e.g. `arr[i], pop();`
  - *lib-unknowns*  
    - e.g. `lib();`
  - *p-unknowns*

- **Cause point = Unknown + program point + unknown-type**
Ranking of Cause Points

1. Unknown types

   \[\text{input} > \text{library} > \text{path} > \text{loop} > \text{computational} > \text{data-structure}\]

2. Grouping of Cause points
   
   A. \textit{Lexical Similarity-based Grouping} \quad \text{e.g. malloc();}
   
   B. \textit{Proximity-based Grouping} \quad \text{e.g. same file or function}

3. Contribution score-based ranking

   \[tc\text{-}\text{score}(cp) = k \times fc\text{-}\text{score}(cp) + pc\text{-}\text{score}(cp)\]
Approach Benefits

**Manual review effort reduction**

- **Reviewing** as many as **alarms** possible in a given time
- **Identifying** as many as **errors** possible in a given time
Experimental Evaluation

**RQ1:** What is the reduction in the manual effort using the proposed approach?

**RQ2:** What is the contribution of cause points in generating the alarms and how are they distributed in practice?

**RQ3:** How effective are the metrics used in the ranking of cause points?

- Implementation using TCS ECA analysis framework
### Effort Reduction (RQ1)

<table>
<thead>
<tr>
<th>App.</th>
<th>Tool used</th>
<th>Verification Property &amp; alarms</th>
<th>Reviewers</th>
<th>Manual Effort (Hrs)</th>
<th>% εsaved</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Original</td>
<td>Proposed</td>
<td>εorig</td>
</tr>
<tr>
<td>A1(C)</td>
<td>TCS ECA</td>
<td>AIOB(215)</td>
<td>R1#</td>
<td>R3#</td>
<td>2.41</td>
</tr>
<tr>
<td></td>
<td>TCS ECA</td>
<td>AIOB (215)</td>
<td>R6</td>
<td>R6</td>
<td>6.83</td>
</tr>
<tr>
<td></td>
<td>TCS ECA</td>
<td>AIOB+OFUF+ZD(1000)</td>
<td>R2#</td>
<td>R2#</td>
<td>9.15</td>
</tr>
<tr>
<td>A2(C)</td>
<td>TCS ECA</td>
<td>AIOB(196)</td>
<td>R5 + R6</td>
<td>R5 + R6</td>
<td>3.29</td>
</tr>
<tr>
<td>A3(C)</td>
<td>TCS ECA</td>
<td>AIOB+ZD (243)*</td>
<td>R7</td>
<td>R8</td>
<td>12.15</td>
</tr>
<tr>
<td>A4(C)</td>
<td>TCS ECA</td>
<td>IDP (2000)*</td>
<td>R1</td>
<td>R2</td>
<td>2.74</td>
</tr>
<tr>
<td>A5</td>
<td>Polyspace</td>
<td>AIOB+ZD (85)</td>
<td>R4</td>
<td>R4</td>
<td>3.53</td>
</tr>
<tr>
<td>(C++)</td>
<td>Code Prover</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>NPEDetector</td>
<td>NDP (555)*</td>
<td>R3#</td>
<td>R2#</td>
<td>5.68</td>
</tr>
</tbody>
</table>
Distribution/Contribution (RQ2)
### Ranking Effectiveness (RQ3)

**Error cause rate (t) =**

\[
\text{Errors uncovered in alarms caused by Cause Points of type } t = \frac{\text{Total alarms caused by Cause Points of type } t}{\text{Total alarms caused by Cause Points of type } t}
\]

<table>
<thead>
<tr>
<th>Cause point (unknown) types</th>
<th>inputs</th>
<th>Library</th>
<th>Path</th>
<th>Loop</th>
<th>Computation</th>
<th>Data Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of cause points</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>16</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Alarms caused</td>
<td>11</td>
<td>7</td>
<td>6</td>
<td>56</td>
<td>12</td>
<td>21</td>
</tr>
<tr>
<td>Errors caused</td>
<td>11</td>
<td>7</td>
<td>3</td>
<td>20</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>% Error cause Rate</td>
<td>100</td>
<td>100</td>
<td>50</td>
<td>35</td>
<td>0</td>
<td>9</td>
</tr>
</tbody>
</table>
Summary

Motivation

Simplifying manual inspections of alarms

Proposed Approach

Cause point Analysis

Intensive static analysis

Iterative static analysis

Thank You!

Approach Benefits

Manual review effort reduction

Reviewing as many as alarms possible in a given time

Identifying as many as errors possible in a given time

Experimental Evaluation

RQ1: What is the reduction in the manual effort using the proposed approach?
  Answer: 42%

RQ2: What is the contribution of cause points in generating the alarms and how are they distributed in practice?
  Answer: 80-20 principle in causing the alarms

RQ3: How effective are the metrics used in the ranking of cause points?
  Answer: Validated our hypothesis in cause points ranking