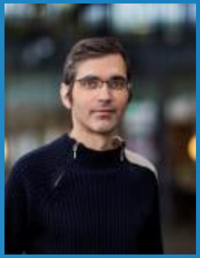
TATA CONSULTANCY SERVICES
Experience certainty.

Reducing Static Analysis Alarms based on Non-impacting Control Dependencies

Tukaram Muske
TRDDC,
Tata Consultancy Services,
Pune, India
t.muske@tcs.com

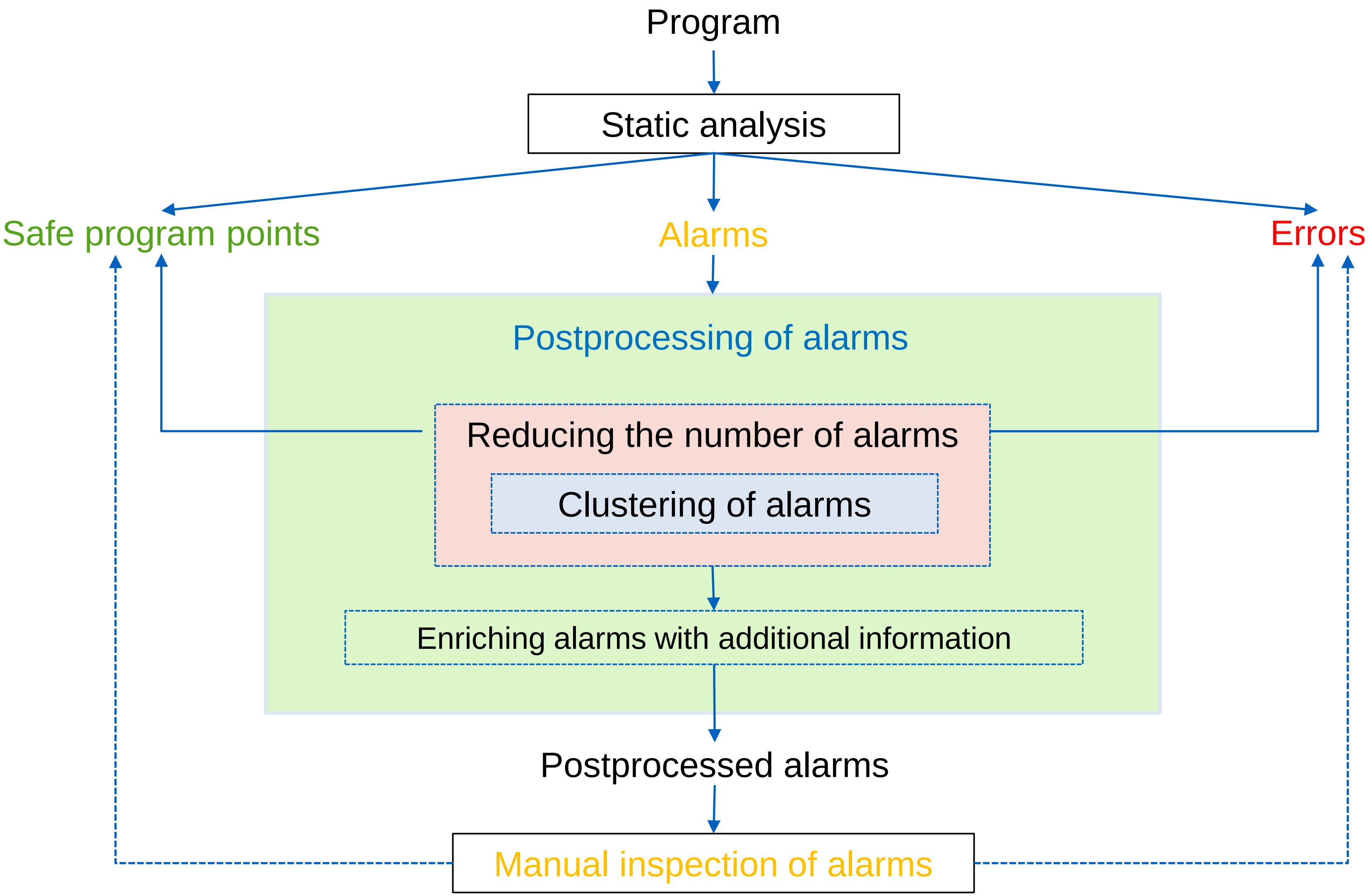
Rohith Talluri
TRDDC,
Tata Consultancy Services,
Pune, India
rohith.talluri@tcs.com



Alexander Serebrenik
Eindhoven University of Technology,
Eindhoven, The Netherlands
a.serebrenik@tue.nl

17th Asian Symposium on Programming Languages and Systems (APLAS)
Nusa Dua, Bali, 2 - 4 December 2019

Background



TATA CONSULTANCY SERVICES
Experience certainty.

Clustering of alarms (Background)

- **Example 1**

A_4 is FP $\Rightarrow$ A_6 is FP

```
void foo(){
  int arr[5], tmp, i = 0;
   …
  arr[i] = 0;   // Dominant alarm
  if(i < tmp){
    arr[i] = 1;// Follower alarm
}
```

A_4 (Dominant alarm), A_6 (Follower alarm)

- **Example 2 (Limitation)**

A_6 is FP $\not\Rightarrow$ A_8 is FP

A_8 is FP $\not\Rightarrow$ A_6 is FP

```
void foo(){
  int arr[5], tmp, i = 0;
  …

  if(i < tmp){
    arr[i] = 0;// Dominant alarm
  else
    arr[i] = 1;// Dominant alarm
}
```

A_6 (Dominant alarm), A_8 (Dominant alarm)

Repositioning of alarms[1] (Background)

- **Example 1**

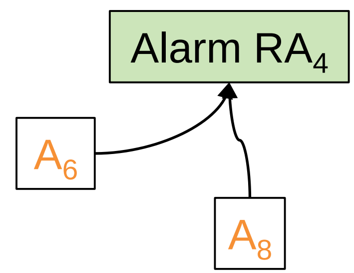
A_4 is FP $\Rightarrow$ A_6 is FP

```
void foo(){
  int arr[5], tmp, i = 0;
   …
  arr[i] = 0;   // Dominant alarm      A4
  if(i < tmp){
    arr[i] = 1;// Follower alarm       A6
}
```

- **Example 2 (**Overcoming limitation of clustering techniques**)**

RA_4 is FP $\Longleftrightarrow$ A_6 and A_8 are FPs

```
void foo(){
  int arr[5], tmp, i = 0;
  …
  //assert(0 ≤ i ≤ 4);
  if(i < tmp){
    arr[i] = 0;
  else
    arr[i] = 1;
}
```



[1] Tukaram Muske, Rohith Talluri, and Alexander Serebrenik. "*Repositioning of static analysis alarms*". In ACM SIGSOFT international symposium on software testing and analysis (ISSTA), pages 187 -197, 2018. ACM

Repositioning of alarms (Background)

- Limitation of the repositioning technique
  - Conservative assumption about the controlling conditions of alarms

- **Limitation Case**

```
void foo(){
    int arr[5], i;
    …

    if(c1)
        arr[i] = 0;

    if(c2)
        arr[i] = 1;
}
```

RA

A_6

A_9

There doesn't exist RA such that

$$RA \text{ is FP } \Longleftrightarrow A_6 \text{ and } A_9 \text{ are FPs}$$

(Because, A_6 can be safe due to `c1`, and A_9 can be safe due to `c2`)

Pilot Study

- What percentage of similar alarms appear in the limitation cases?

- Study using
  - 64779 alarms on 16 open source applications
  - For 5 verification properties – AIOB, DZ, OFUF, IDP, and UIV
  - Resulting after their repositioning

- Results
  - 50% of alarms are similar
  - Alarms in the limitation cases
    - 74% of the similar alarms
    - 38% of the total alarms

Considerable number of similar alarms are not grouped together due to the conservative assumption!

Our Solution - Overview

- Introduce notion of
  - non-impacting control dependencies (NCDs) of alarms
  - Impacting control dependencies (ICDs) of alarms
- Compute approximated NCDs/ICDs
- Use them to improve alarms repositioning

- **Motivating Example**

If

$n_5 \rightarrow n_6$ (i.e. `c1`) is NCD of A_6

and

$n_8 \rightarrow n_9$ (i.e. `c2`) is NCD of A_9

Then,

RA_4 is FP $\Longleftrightarrow$ A_6 and A_9 are FPs

```
1.    void foo(){
2.        int arr[5], i = 0;
3.        …
4.       //assert(0 ≤ i ≤ 4);
5.        if(c1)
6.            arr[i] = 0;
7.
8.        if(c2)
9.            arr[i] = 1;
10.   }
```

Alarm RA_4 (line 4); A_6 (line 6); A_9 (line 9)

The Notion of ICDs/NCDs

Program P

```
1.  void foo(){
2.   …
3.   if(c){
4.    if(…)
5.     arr[i] = 1; // Alarm α
6.   }
7.  }
```

Program P’

```
1.  void foo(){
2.   …
3.   if(nondet()){
4.    if(…)
5.     arr[i] = 1; // Alarm α
6.   }
7.  }
```

A transitive control dependency $n_x \rightarrow n_y$ (e.g. $n_3 \rightarrow n_4$) of α is an ICD only if

1. α is a false positive in P; and
2. P’ s.t. condition of n_x is replaced by nondeterministic choice function, and α is an error in P’

Otherwise, $n_x \rightarrow n_y$ is an NCD of α.

Illustrating NCDs

Case 1

```
void foo(){
  i = safeValues();

    if(c){
      if(…)
        arr[i];// α
    }
}
```

$n_5 \rightarrow n_6$ is NCD of α.

Case 2.1

```
void foo(){
  i = unafeValues();

    if(c){
      if(…)
        arr[i];// α
    }
}
```

The unsafe values reach α. Then,

$n_5 \rightarrow n_6$ is NCD of α

Case 2.2

```
void foo(){
  i = unafeValues();

    if(c){
      if(…)
        arr[i]; // α
    }
}
```

The unsafe values **do not reach** α due to "C". Then,

$n_5 \rightarrow n_6$ is ICD of α

But, how to compute NCDs/ICDs of alarms?

Approximated NCDs

- Observation
  - A control dependency rarely makes an alarm safe (safety condition)
  - Value Slice [2]
    - Transitive control dependencies of alarms rarely make the alarms safe (2% of alarms)
- *Intuitively, the chance of existing different safety condition for each of the similar alarms is even lower.*

$n_5 \rightarrow n_6$ (i.e. `c1`) is NCD of A_6

and

$n_8 \rightarrow n_9$ (i.e. `c2`) is NCD of A_9

```
1.  void foo(){
2.      int arr[5], tmp = 1, i = 0;
3.      …
4.      //assert(0 ≤ i ≤ 4);      Alarm RA4
5.      if(c1)
6.          arr[i] = 0;           A6
7.
8.      if(c2)
9.          arr[i] = 1;           A9
10. }
```

[2] Amitabha Sanyal, and Uday P. Khedker. *"Value slice: A new slicing concept for scalable property checking".* In International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), pp. 101-115. Springer, Berlin, Heidelberg, 2015.

NCD-based Repositioning

Let Θ_s be a set of similar alarms, and R be the set of alarms after their repositioning

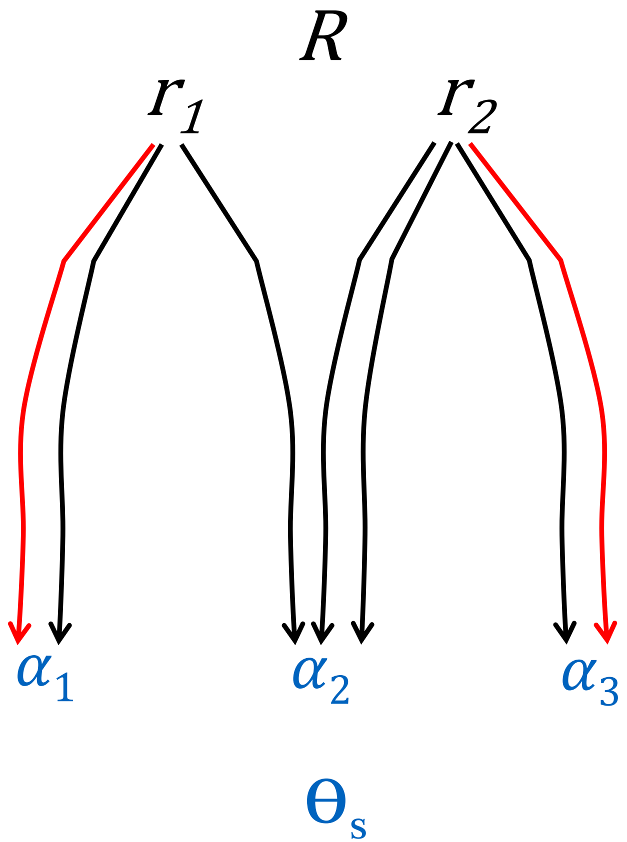
- **Constraint 1** (Safety)

  Program points of the repositioned alarms R together dominate the program point of every alarm in Θ_s

- **Constraint 2** (overcoming spurious error detection)
- For every repositioned alarm *r* in *R*, there exists a path between *r* and $\phi \epsilon \Theta_s$ such that the path does not have an ICD of ϕ.

- **Constraint 3**

  The number of the repositioned alarms R is strictly not greater than the number of original alarms Θ_s



Repositioning Technique

- Data flow analysis-based technique
  - Computing approximated NCDs
  - Repositioning with the three constraints
  - Postprocessing of repositioned alarms
  - More details in the paper

Evaluation

- Implementation
  - Analysis framework of TCS ECA
  - Limited inter-functional repositioning

- 105,546 alarms generated on
  - 32 applications
    - 16 open source
    - 16 Industry (11 C and 5 COBOL)
  - 5 verification properties
    - AIOB, DZ, OFUF, IDP and UIV
  - Resulting after state-of-the-art grouping and repositioning

Evaluation Results

Application category	Max. reduction	Average reduction	Median reduction
Open Source	23.57%	10.16%	9.02%
C Industry	29.77%	8.97%	17.18%
COBOL Industry	36.09%	27.68%	28.61%

- Evaluation of spurious error detection
  - Manual analysis of 150 repositioned alarms
  - Corresponding to 482 original alarms
  - Reduction 70% with spurious error detection rate 2%

Summary

Problem

- Around 38% of the alarms still are not grouped by State-of-the-art alarms clustering and repositioning techniques

Our solution

- Introduced the notion of NCDs of alarms
- Computation of approximated NCDs
- NCD-based repositioning

Technique

- Data-flow analysis based technique
- Performing NCDs computation and repositioning together

Evaluation

Application category	Max. reduction	Average reduction	Median reduction
Open Source	23.57%	10.16%	9.02%
C Industry	29.77%	8.97%	17.18%
COBOL Industry	36.09%	27.68%	28.61%

- Safe reduction, however spurious error detection rate of 2%