STIC Doctoral School of the university of Nice Sophia Antipolis
I3S laboratory

Constraint-based fault localization

Field: *Computer Science*
EJCP 2013

Presented by:
Mohammed Bekkouche
Cooperation with :
Michel Rueher, supervisor
Hélène Collavizza, co-supervisor
Yahia Lebbah
Olivier Ponsini

Abstract

Abstract

- Error Localization, Software Debugging, Software Engineering

Abstract

- Error Localization, Software Debugging, Software Engineering

- A counterexample -> Faulty execution trace of the counterexample

Abstract

- Error Localization, Software Debugging, Software Engineering

- A counterexample -> Faulty execution trace of the counterexample
- The constraint programming formalism
  Why ?
  - To model the problem,
  - And to solve it.

Abstract

- Error Localization, Software Debugging, Software Engineering

Work objective

- Locate suspicious instructions in imperative programs
- For which a counterexample has been found with Bounded Model Checker(BMC) tool

- A counterexample -> Faulty execution trace of the counterexample
- The constraint programming formalism
  Why ?
  - To model the problem,
  - And to solve it.

Problem

Problem

- A program may contain errors
- This errors can harm in proper operation of the program

Problem

- A program may contain errors
- This errors can harm in proper operation of the program
- The process of software debugging is essential
  - errors detection, **faults localization**, correction of fautes

Problem

- A program may contain errors
- This errors can harm in proper operation of the program
- The process of software debugging is essential
  - errors detection, **faults localization**, correction of fautes
- Program with errors :
  - A tool for model-checking (e.g. CPBPV, CBMC) to obtain a counterexample
  - Counterexample -> excution trace of counterexample

Problem

- A program may contain errors
- This errors can harm in proper operation of the program
- The process of software debugging is essential
  - errors detection, **faults localization**, correction of fautes
- Program with errors :
  - A tool for model-checking (e.g. CPBPV, CBMC) to obtain a counterexample
  - Counterexample -> excution trace of counterexample
- The problem :
  - The execution trace of the counterexample is often long and difficult to understand

Problem

- A program may contain errors
- This errors can harm in proper operation of the program
- The process of software debugging is essential
  - errors detection, **faults localization**, correction of fautes
- Program with errors :
  - A tool for model-checking (e.g. CPBPV, CBMC) to obtain a counterexample
  - Counterexample -> excution trace of counterexample
- The problem :
  - The execution trace of the counterexample is often long and difficult to understand

Our idea :

- Counterexample, program and the postcondition -> set of infeasible constraints -> A **minimal conflict set of constraints** (**IIS**)

```
1  class program{
2
3   /*@ ensures
4     @ (c >= d+e);
5     @*/
6  void foo(int a,int b){
7    int c;
8    int d;
9    int e;
10   int f;
11   if (a>=0){
12     c=a;
13     d=a;
14     e=b;
15   }
16   else{
17     c=b; /* error */
18     d=1;
19     e=-a;
20     if (a>b){
21       f=b+e+a;
22       d=d+4;
23     }
24     else{
25       f=e;
26     }
27   }
28   c=c+d+e;
29 }
30 }
```

Motivation example

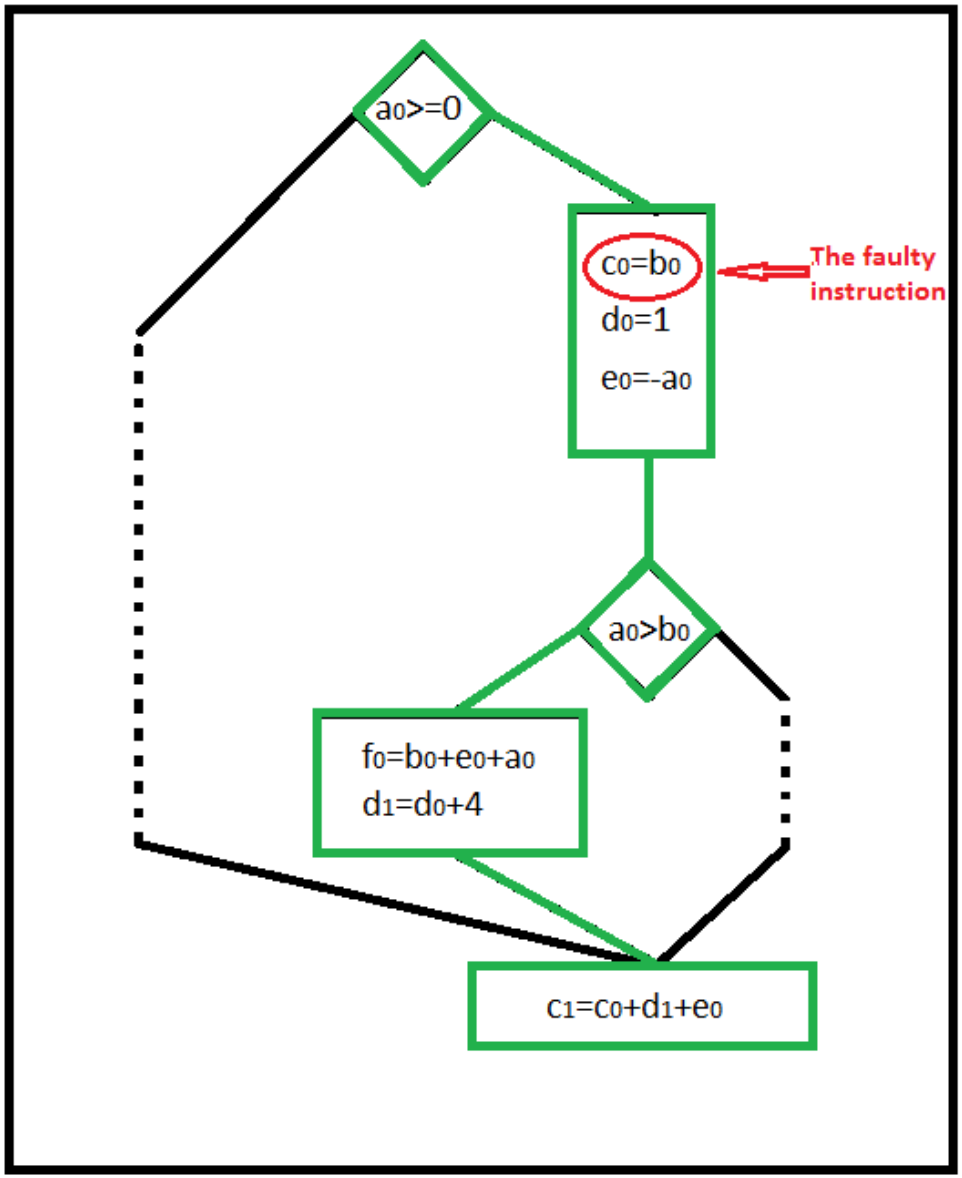

FIGURE: The control flow graph of the foo program in DSA form

Motivation example

Approche to locate faults :

Motivation example

Approche to locate faults :

- Use a BMC tool to obtain a counterexample :
  CE_{PROG} $(a_0 = -1, b_0 = -2)$

Motivation example

Approche to locate faults :

- Use a BMC tool to obtain a counterexample :
  CE_{PROG} $(a_0 = -1, b_0 = -2)$
- Generate the set of constraints which corresponds to the trace of the counterexample :
  $C_{TCE} = \{c_0 = b_0, d_0 = 1, e_0 = -a_0, a_0 > b_0, f_0 = b_0 + e_0 + a_0, d_1 = d_0 + 4, c_1 = c_0 + d_1 + e_0\}$
- Generate the constraints set that corresponds to the postcondition :
  $C_{POST} = \{c_1 >= d_1 + e_0\}$
- Generate the constraints set of the counterexample :
  $C_{CE_{PROG}} = \{a_0 = -1, b_0 = -2\}$

Motivation example

Approche to locate faults :

- Identification of the faulty contraints :

Motivation example

Approche to locate faults :

- Identification of the faulty contraints :
  - $C = C_{CE_{PROG}} \cup C_{TCE} \cup C_{POST}$ is infeasible
    $\Rightarrow$ It has at least an infeasible sub-system irreducible of constraints (**IIS**)

Motivation example

Approche to locate faults :

- Identification of the faulty contraints :
  - $C = C_{CE_{PROG}} \cup C_{TCE} \cup C_{POST}$ is infeasible
    $\Rightarrow$ It has at least an infeasible sub-system irreducible of constraints (**IIS**)
  - C_{LOC} must be infeasible and minimum
    $C_{LOC} = \{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is infeasible
    - $\{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is feasible
    - $\{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is feasible
    - $\{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is feasible
    - $\{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is feasible

Motivation example

Approche to locate faults :

- Identification of the faulty contraints :
  - $C = C_{CE_{PROG}} \cup C_{TCE} \cup C_{POST}$ is infeasible
    $\Rightarrow$ It has at least an infeasible sub-system irreducible of constraints (**IIS**)
  - C_{LOC} must be infeasible and minimum
    $C_{LOC} = \{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is infeasible
    - $\{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is feasible
    - $\{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is feasible
    - $\{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is feasible
    - $\{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is feasible
  - $C' = (C_{CE_{PROG}} \cup C_{TCE} \cup C_{POST}) \backslash c_i$ is feasible
    $(c_i \in C_{LOC})$ Because the input infeasible system has a single **IIS**

Motivation example

Approche to locate faults :

- Identification of the faulty contraints :
  - $C = C_{CE_{PROG}} \cup C_{TCE} \cup C_{POST}$ is infeasible
    $\Rightarrow$ It has at least an infeasible sub-system irreducible of constraints (**IIS**)
  - C_{LOC} must be infeasible and minimum
    $C_{LOC} = \{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is infeasible
    - $\{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is feasible
    - $\{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is feasible
    - $\{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is feasible
    - $\{b_0 = -2, c_0 = b_0, c_1 = c_0 + d_1 + e_0, c_1 >= d_1 + e_0\}$ is feasible
  - $C' = (C_{CE_{PROG}} \cup C_{TCE} \cup C_{POST}) \backslash c_i$ is feasible
    $(c_i \in C_{LOC})$ Because the input infeasible system has a single **IIS**
- $LOC = \{$***ligne 17***$, ligne\ 28\}$

Approach

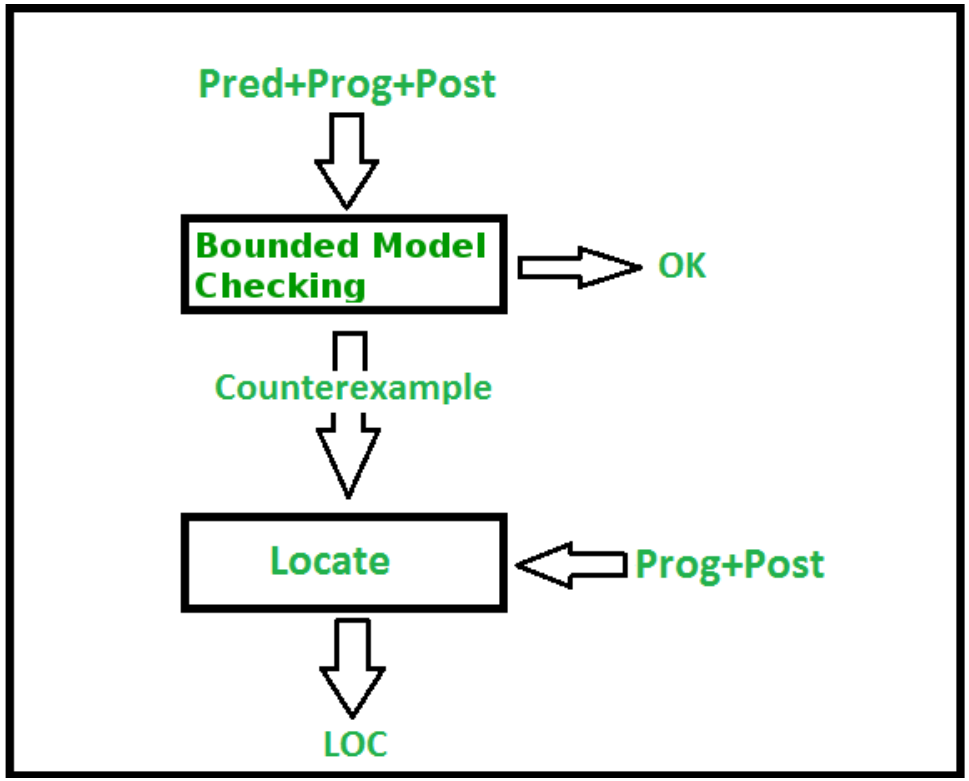

FIGURE: Our approach of localization

Implementation

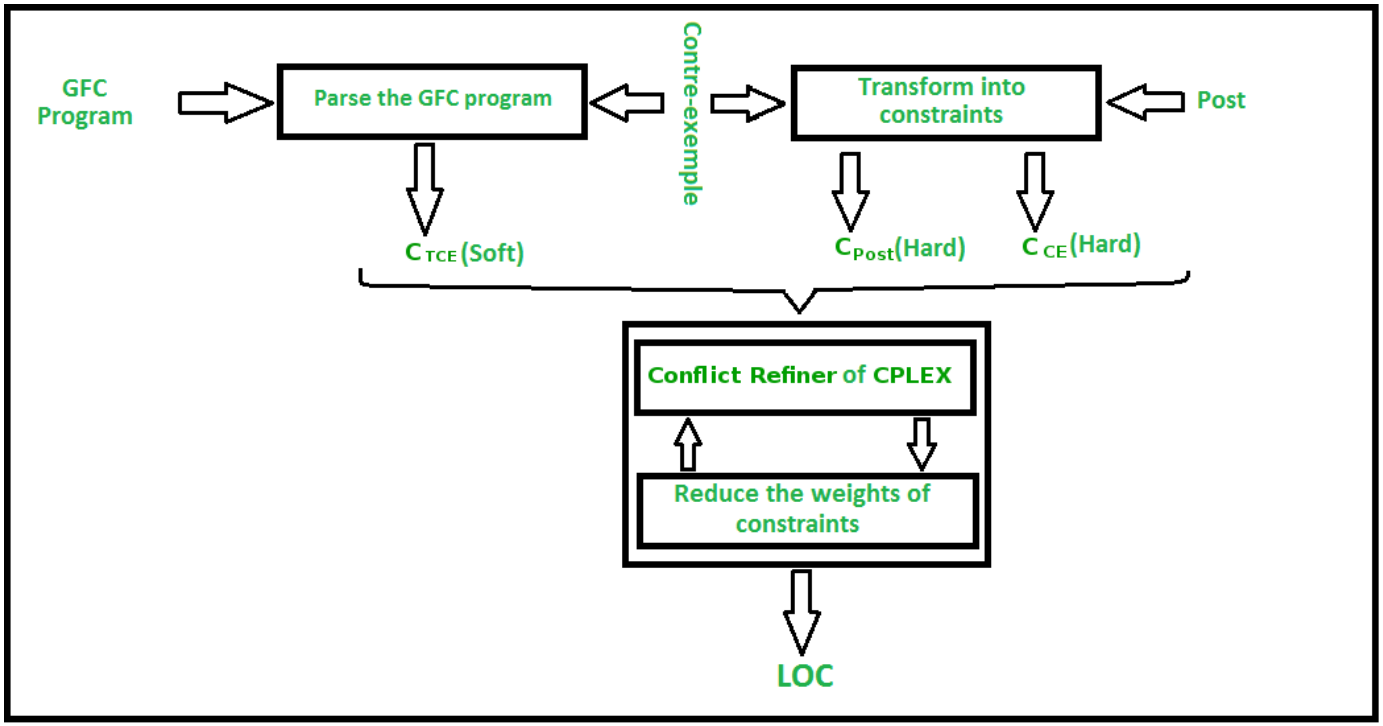

FIGURE: The localization process

Thank you for your attention