Faulty Interaction Identification via Constraint Solving and Optimization

Jian Zhang, **Feifei Ma** and Zhiqiang Zhang

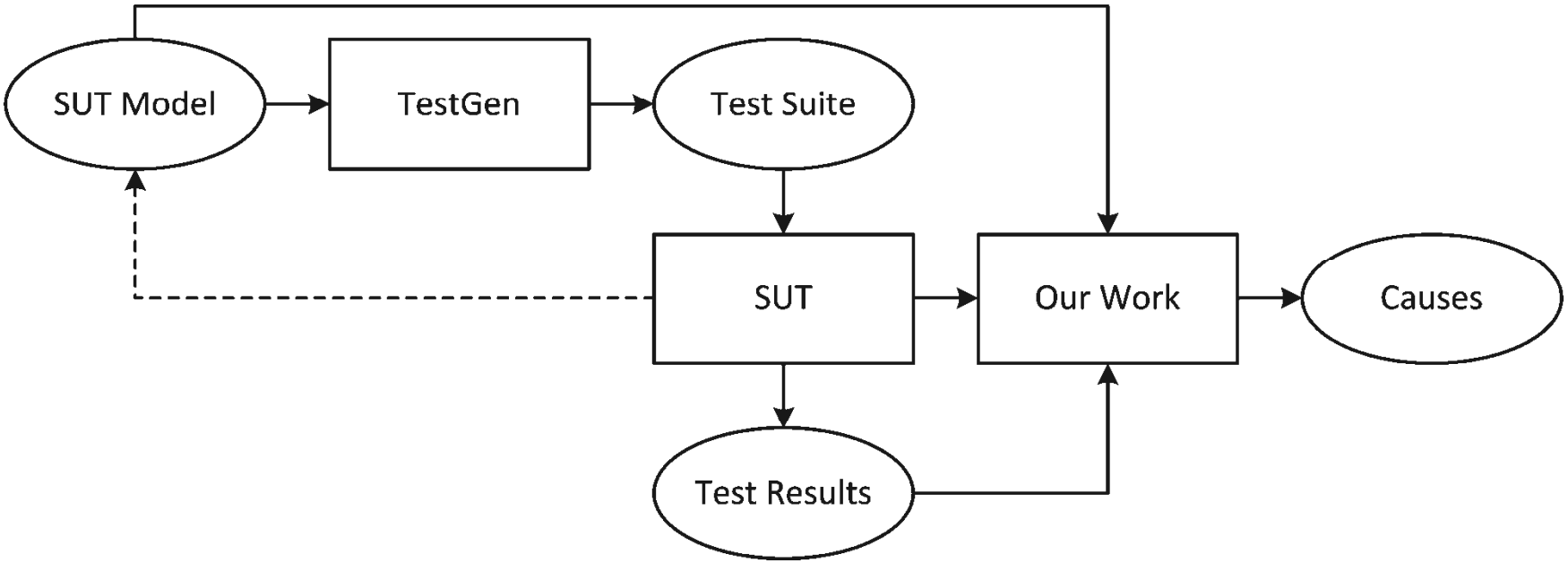
Outline

- Introduction
- Problem translation
- Experimental results
- Conclusions

Introduction

Software testing and debugging

- Test case generation / test suite design
- Test suite execution
- Fault localization / debugging


SUT Model
TestGen
Test Suite
SUT
Our Work
Causes
Test Results

Combinatorial testing

- Combinatorial testing is an effective method to test parameterized systems
- Adopted by AT&T, IBM, Motorola, Microsoft, …
- Combinatorial testing assumes that failures are caused by certain parameter interactions (faulty combinatorial interactions, *FCI*s).

Covering Arrays

- For combinatorial testing, we usually use covering array (CA) to represent the test suite.
- A CA is an array, such that each row represents a test case, and each column corresponds to a parameter.
- A CA of strength *t* covers all parameter value combinations between all *t* parameters.

An Example

- Suppose we are testing an online payment system. It has 4 parameters of level 3.
- The covering array is as follows:

Client	Server	Payment	Database	Exec Result
Firefox	WebSphere	MasterCard	DB2	pass
Firefox	.NET	UnionPay	Oracle	pass
Firefox	Apache	Visa	Access	fail
IE	WebSphere	UnionPay	Access	pass
IE	Apache	MasterCard	Oracle	fail
IE	.NET	Visa	DB2	pass
Opera	WebSphere	Visa	Oracle	pass
Opera	.NET	MasterCard	Access	pass
Opera	Apache	UnionPay	DB2	pass

- – Client can be: 1. Firefox; 2. IE; 3. Opera.
- – WebServer can be: 1. WebSphere; 2. .NET; 3. Apache.
- – Payment can be: 1. MasterCard; 2. UnionPay; 3. VISA.
- – Database can be: 1. DB/2; 2. Oracle; 3. Access.

The Problem

- Two test cases fail due to two FCIs respectively.
- Suppose we don't know the FCIs in advance.
  It is not easy to tell what the FCIs are just from the test results.
- The problem is:
  - How to identify the FCIs from the test result of the CA?
    (i.e. post-analysis techniques)

CSP Formulation

- Suppose a SUT (System Under Test) :
  - has *k* attributes/parameters/components.
  - the i^{th} attribute has domain D_i = {1,2, …}.
- Suppose there are *m* test cases, some passed, some failed.
- Find out a set of FCIs, such that a test case fails iff it matches at least one of the FCIs.

One FCI

- Suppose the FCI is $<x_1, x_2, \ldots x_k>$. $x_i : \{0\} \cup D_i$
- $x_i = 0$ means the j_{th} attribute is not in the FCI.

➢ For each failing test case $(w_1, w_2, \ldots, w_k)$, add a constraint

$((x_1=0) \text{ OR } (x_1=w_1))$
AND $((x_2=0) \text{ OR } (x_2=w_2))$
AND .....
AND $((x_k=0) \text{ OR } (x_k=w_k))$

➢ For each passing test case $(v_1, v_2, \ldots, v_k)$, add a constraint

$(\text{NOT}(x_1=0) \text{ AND } \text{NOT}(x_1=v_1))$
OR $(\text{NOT}(x_2=0) \text{ AND } \text{NOT}(x_2=v_2))$
OR .....
OR $(\text{NOT}(x_k=0) \text{ AND } \text{NOT}(x_k=v_k))$

Two FCIs

- Two FCIs: $\langle x_1, x_2, \ldots x_k \rangle$ and $\langle y_1, y_2, \ldots y_k \rangle$
- For each failing test case $(w_1, w_2, \ldots, w_k)$, add:

$$
\begin{aligned}
&\quad (((x_1=0) \ \text{OR} \ (x_1=w_1)) \\
&\text{AND} ((x_2=0) \ \text{OR} \ (x_2=w_2)) \\
&\text{AND} \ \ldots\ldots \\
&\text{AND} ((x_k=0) \ \text{OR} \ (x_k=w_k))) \\
&\textbf{OR} \\
&\quad (((y_1=0) \ \text{OR} \ (y_1=w_1)) \\
&\text{AND} ((y_2=0) \ \text{OR} \ (y_2=w_2)) \\
&\text{AND} \ \ldots\ldots \\
&\text{AND} ((y_k=0) \ \text{OR} \ (y_k=w_k)))
\end{aligned}
$$

Multiple FCIs

- For each passing test case (v_1, v_2, …, v_k) , add :

  ((NOT(x_1=0) AND NOT(x_1=v_1))
  OR (NOT(x_2=0) AND NOT(x_2=v_2))
  OR .....
  OR (NOT(x_k=0) AND NOT(x_k=v_k)))

  AND

  ((NOT(y_1=0) AND NOT(y_1=v_1))
  OR (NOT(y_2=0) AND NOT(y_2=v_2))
  OR .....
  OR (NOT(y_k=0) AND NOT(y_k=v_k)))

Symmetry Breaking

- Symmetries are introduced by permutations of FCIs.

  E.g. *If* $\{x_1=v_1, x_2=v_2, x_3=v_3, x_4=v_4\}$ *and* $\{y_1=v_1', y_2=v_2', y_3=v_3', y_4=v_4'\}$ *is a solution, so is* $\{x_1=v_1', x_2=v_2', x_3=v_3', x_4=v_4'\}$ *and* $\{y_1=v_1, y_2=v_2, y_3=v_3, y_4=v_4\}$.

➢ To break permutation symmetries, lexicographical ordered constraints are enforced.

((x1<=y1))

AND ((x1<y1) OR (x2<=y2))

AND ((x1<y1) OR (x2<y2) OR (x3<=y3))

AND ((x1<y1) OR (x2<y2) OR (x3<y3) OR (x4<y4))

Translation to PBO

- There might be many solutions to the FCI identification problem, we need to find the optimal one.

  E.g. *If {$x_2=v_2$, $x_3=v_3$} is a FCI, then {x1=1, $x_2=v_2$, $x_3=v_3$} and {$x_2=v_2$, $x_3=v_3$, x4=4} are also FCIs.*
- Investigate FCI identification as an optimization problem instead of a decision problem.
- The problem can be formulated as a Pseudo Boolean Optimization Problem.

Pseudo-Boolean Optimization

- Linear PB constraint: $\sum_i a_i x_i \geq b$

  where x_i is a Boolean variable, a_i, b are integers.
- A PB constraint is nonlinear if it contains the product of Boolean variables.

$$\sum_i a_i \left(\prod_k x_{i,k} \right) \geq b$$

- A PBO Problem is to maximize (minimize) a PB expression subject to a set of PB constraints.

PBO Encoding

- Suppose there are *w* failing cases for the SUT, and we are going to find *n (n ≤ w)* FCIs.
- Boolean variables:

  - Primary variables

    $P_{i,j,v} \equiv (x_{i,j} = v)$, i.e., the j^{th} parameter of the i^{th} FCI x_i takes value *v. (1 ≤ j ≤ k, 1 ≤ i ≤ n)*

  - Auxiliary variables

    $E_{t,i}$: The t^{th} failing case is caused by the i^{th} FCI.

PBO Encoding

- Objective: Minimize the size of FCIs / Maximize the number of '0's of all FCIs.

$$Minimize \quad -\sum_i \sum_j P_{i,\ j,0}$$

- PB Constraints:

1. Basic Constraints: guarantee the validity of the encoding.

for all $1 \le i \le n$, $1 \le j \le k$, add

$$\sum_v P_{i,\ j,\ v} = 1 \qquad \text{where } v \in D_i \cup \{0\}$$

Each parameter of each FCI can take only one value.

2. Constraints for passing cases:

   Remove the innermost AND operator in the CSP.

   E.g. Suppose $D_1 = 3$, we replace (NOT(x_1=0) AND NOT(x_1=1)) with (x_1=2 OR x_1=3)

   for each passing case V = (v_1, v_2, …, v_k) , we add

$$\bigwedge_{1 \le i \le n} \Sigma_{j=1}^{k} \Sigma_{v, v \neq 0, v \neq v_j} P_{i,j,v} \geq 1$$

3. Constraints for failing cases

- One FCI

for each failing case V = (v1, v2, …, vk) , we add

$$\bigwedge_{1 \le j \le k} P_{1,j,0} + P_{1,j,v_j} = 1$$

- Multiple FCIs

a) Each failing case must match at least one FCI.

for 1 ≤ t ≤ w

$$\sum_{i=1}^{n} E_{t,i} \ge 1$$

b) Each FCI must match at least one failing case.

for $1 \le i \le n$ $\quad \sum_{t=1}^{w} E_{t,i} \ge 1$

c) If the t^{th} failing case matches the i^{th} FCI, then the j^{th} parameter of the i^{th} FCI either takes value 0 or takes value v_j, and vise versa.

$$\bigwedge_{1 \le i \le n} \bigwedge_{1 \le j \le k} -E_{t,i} + P_{i,j,0} + P_{i,j,v_j} \ge 0$$

And

$$\bigwedge_{1 \le i \le n} E_{t,i} + \Sigma_{1 \le j \le k} \Sigma_{v, v \ne 0, v \ne v_j} P_{i,j,v} \ge 1$$

4. Symmetry Breaking constraints

A direct way to encode the inequality (e.g. x1<y1) is to enumerate all assignments allowed by the inequality and assert that at least one assignment is true.

The i^{th} FCI is lexicographically smaller than the $(i+1)^{th}$ FCI:

$$\left(\bigwedge_{1\leq j\leq k-1} \Sigma_{l=1}^{j}\Sigma_{v_1<v_2}P_{i,l,v1}P_{i+1,l,v2}+\Sigma_v P_{i,j,v}P_{i+1,j,v}\geq 1\right) \bigwedge \Sigma_{l=1}^{k}\Sigma_{v_1<v_2}P_{i,l,v1}P_{i+1,l,v2}\geq 1$$

Experimental Results

- Abstract experiments
  - For given parameters, generate a covering array
  - Set some FCIs in advance, and label test cases matching those FCIs as “fail”
  - Use our method on the labeled test suite to localize the FCIs

Results

Test Suite	nTC	nFCI	sFCI	nFT	nSol/nSol_NSB	tSB (s)	tNSB (s)
$CA(6^{10}, 3)$	526	4	1-4(3/2)	97	1/162	0.177	0.157
		8	1-4(5/1.8)	180	6/-	7.177	NA
$CA(3^{50}, 3)$	133	2	2&3(2/2.5)	20	1/6	0.137	0.052
		4	1-4(3/2)	56	30/-	1.854	NA
$CA(3^{50}, 4)$	579	2	2&3(2/2.5)	83	1/192	0.302	0.279
		4	1-4(4/2.5)	256	1/-	53.185	NA
$CA(3^{100}, 4)$	169	2	2&3(2/2.5)	22	1/6	0.920	0.201
		4	1-4(3/2)	74	1/39366	11.975	51.781

Timeout: 300 seconds

Experiments On A Real System

- Experiment subject:
  - Traffic Collision Avoidance System (TCAS)
- Steps:
  - Build a model for the SUT
  - Generate a covering array from the model
  - Execute test cases on the SUT
  - Use our method on the test results to localize the FCIs

Results

Version	v10			v11			v12		
t	2	3	4	2	3	4	2	3	4
nTC	100	402	1410	100	402	1410	100	402	1410
nFTC	1	7	16	1	9	23	3	14	57
nSol	1	12	700	1	60	NA	1	NA	NA
nSol_NSB	1	72	NA	1	1440	NA	6	NA	NA
nFCI	1	3	7	1	4	≥ 10	3	≥ 8	≥ 10
asFCI	2	5	5.57	2	4.75	-	2	-	-
tSB (s)	0.002	0.045	14.035	0.002	0.227	-	0.244	-	-
tNSB (s)	=	0.049	NA	=	0.865	-	0.257	-	-

Observation

- There may be more than 1 possible solutions. The # of solutions indicates the precision of FCI localization.
- Symmetry breaking introduces extra cost to solving. When # of FCIs is small, the solving time may increase, while when # of FCIs increase, the effect of symmetry breaking emerges
- Solving time increases with # of test cases and # of FCIs

Conclusion

- A new automated approach for identifying FCIs
- Provide a few FCIs which are helpful to the user when debugging
- Future work
  - Different encodings of the problem
  - Other solvers

Thanks!