Empirical Evaluation on FBD Model-Based Test Coverage Criteria using Mutation Analysis

Donghwan Shin, Eunkyoung Jee, and Doo-Hwan Bae

Korea **A**dvanced **I**nstitute of **S**cience and **T**echnology

KAIST Computer Science

KAIST SE LAB

KAIST Software Engineering Laboratory

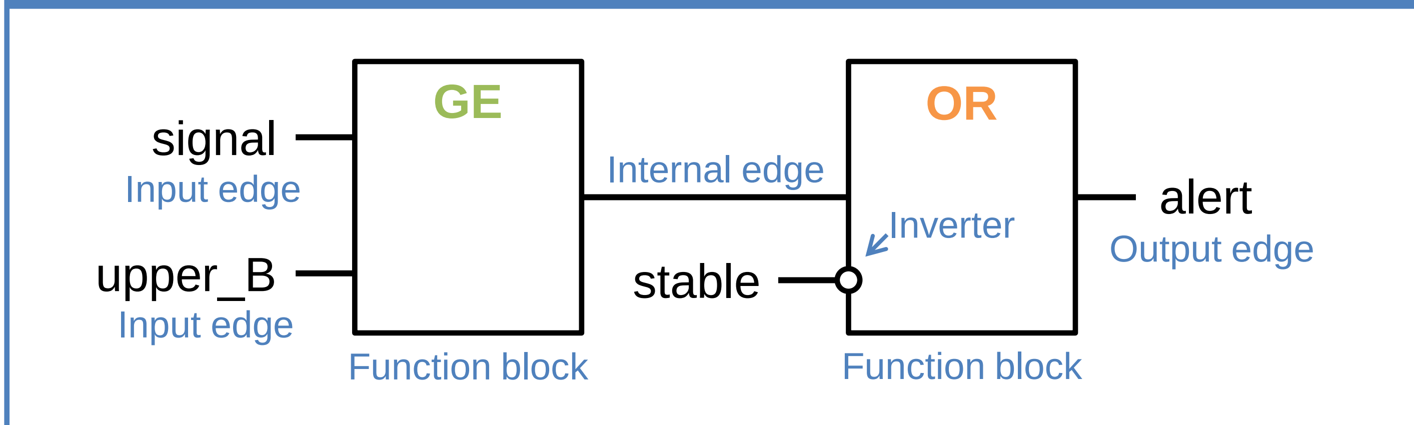
MODELS 2012

innsbruck

Function Block Diagram

- One of PLC (Programmable Logic Controller) languages
  - Dataflow, visual modeling language
  - Model-driven development concepts (C code is automatically generated)

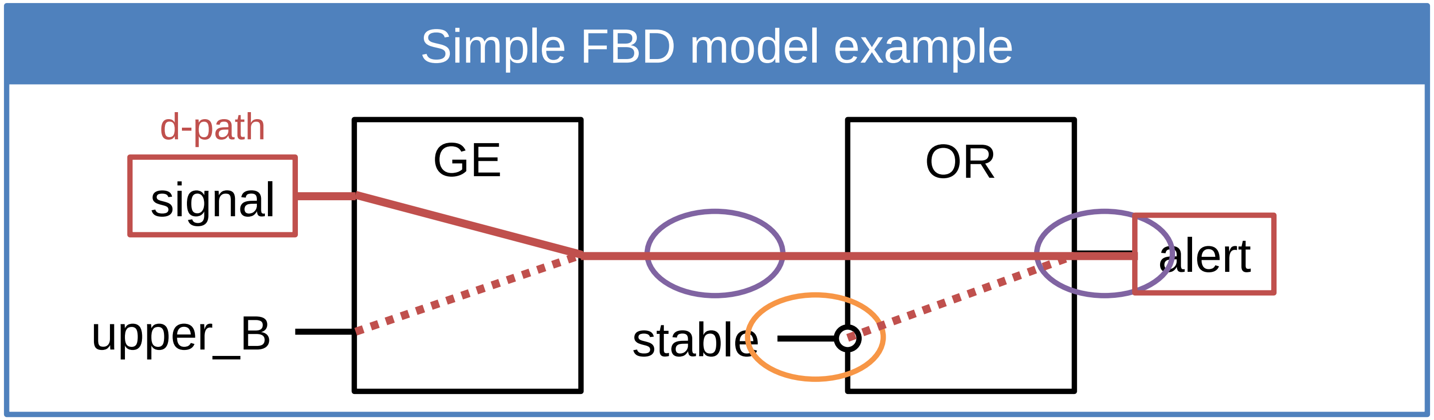
Simple FBD model example



if (signal >= upper_B) OR (not stable) **then** alert

Conventional Wisdom

- Test coverage criterion is a goal or stopping rule for structural testing.
- Three FBD model-based test coverage criteria have been proposed [Jee et al.*]
  - **BC** (Basic Coverage) : every *complete* ***d-path*** must be tested at least once.
  - **ICC** (Input Condition Coverage): BC **+** consider variation of **Boolean input edges**.
  - **CCC** (Complex Condition Coverage): ICC **+** consider variation of **all Boolean edges**.



d-path (BC)

Boolean **Input** edge (ICC)

Boolean **internal**, **output** edge (CCC)

Complete d-path: a finite sequence of edges start from input to ended with output.

* Eunkyoung Jee , Junbeom Yoo , Sungdeok Cha , Doohwan Bae, “A data flow-based structural testing technique for FBD programs”, Information and Software Technology, v.51 n.7, p.1131-1139, July, 2009

Motivation

- FBD test criteria support,
  1. **Dataflow-centric** characteristics of FBD models
  2. **Intuitive** structural coverage concepts for testers

- Still lack of fundamental understanding on,
  1. **Fault detection effectiveness**
  2. **Strong** and **weak points** in terms of fault detection

- We need **systematic investigation** on fault detection of FBD test criteria!

Key Findings

- Q1: Fault detection **effectiveness**
  - CCC detects **up to 97.1% of faults** in the best case
  - In average, **BC**(64.7%) < **ICC**(68.6%) < **CCC**(81.2%)

- Q2: **Strength** and **weakness** on fault detection
  - CCC detects faults in **Boolean edges** and **arithmetic** blocks **over 90%.**
  - **Comparison**, **logic**, and **constants faults** are **not detected up to 18.5%**.

Outline

- Approach Overview
  - Research questions
  - Mutation analysis
  - Study method
- Experimental Description
  - Subject models
  - Test suite generation
  - Mutant generation
- Analysis Results
- Summary and Future Work

Research Questions

- Q1: Fault detection **effectiveness**
  - How effective is each of the three FBD test criteria in fault detection?

- Q2: **Strength** and **weakness** on fault detection
  - What type of faults are likely to be found by the coverage criteria?

Mutation Analysis

- Measures the **fault detection effectiveness** of given test suites.
  - **Mutant score** = (detected mutants / total mutants) * 100

Simple mutation analysis example

Original

```
add(int a, int b) {
    return c = a + b;
}
```

Mutant generation process

Applied mutation operator
= replace arithmetic operator

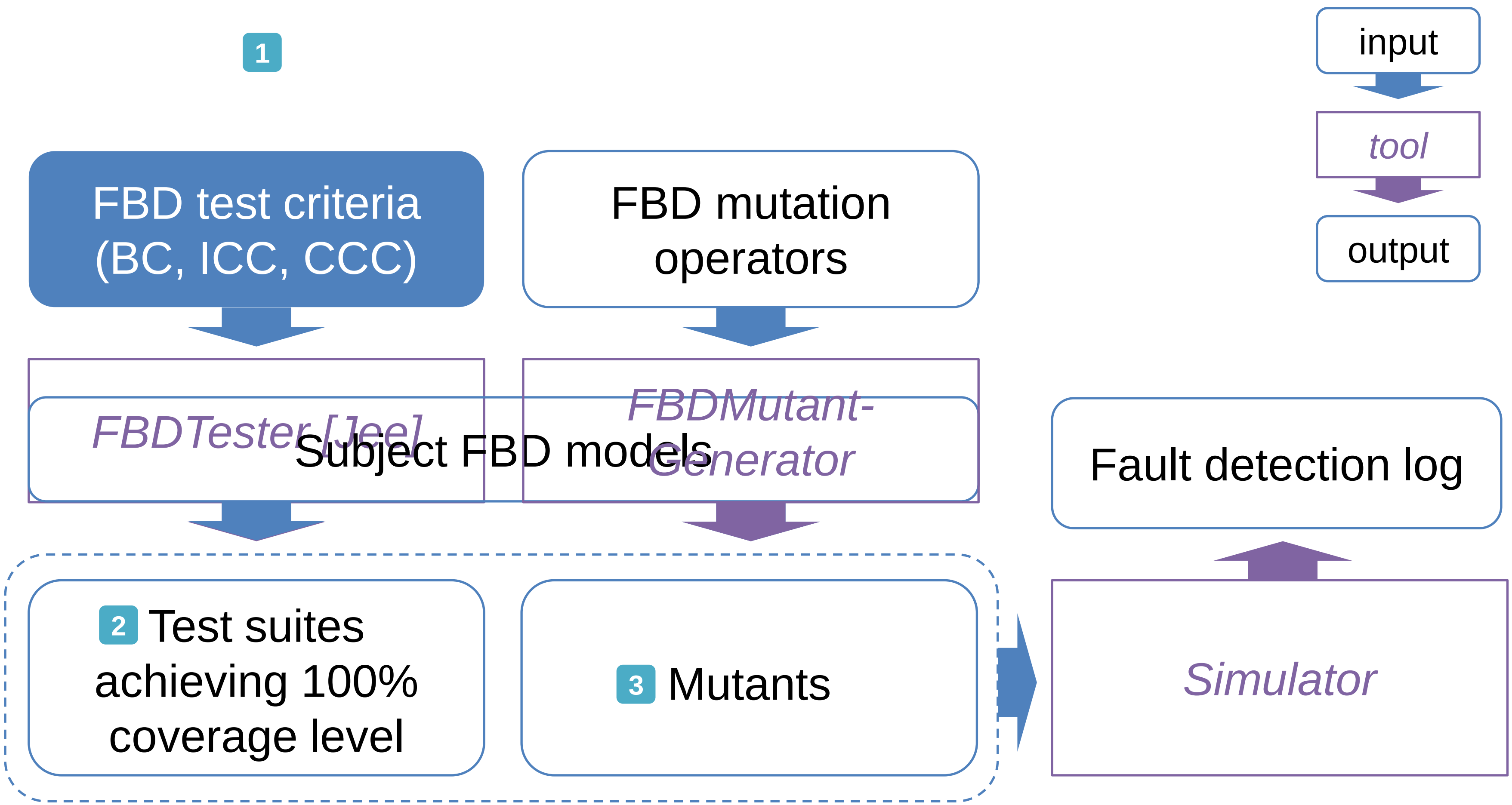
Mutant #1
Mutant #2
...
Mutant #n

```
add(int a, int b) {
    return c = a - b;
}
```

➡ Test case [a=1, b=2]->[c=3] kills (detects) the mutant #n.

- ✓ If a test suite ***S*** kills 80 mutants over 100, **the mutant score** = 80.
- ✓ ***The fault detection effectiveness*** *of* ***S*** *= 80%*

Study Method



Outline

- Approach Overview
  - Research questions
  - Mutation analysis
  - Study method
- Experimental Description
  - Subject models
  - Test suite generation
  - Mutant generation
- Analysis Results
- Summary and Future Work

Subject Models

- Step1: Select subject models
  - Five **real industry FBD models** from KNICS* project** by considering size and structural diversity.

Size information for five subject models

Component	FR	HB	MRC	MRF	TD
Blocks	26	38	15	26	7
d-Paths	142	118	113	235	16
Inputs	30	12	21	30	9
Outputs	4	4	2	4	2
TRs for BC	142	118	113	235	16
TRs for ICC	182	118	165	299	32
TRs for CCC	994	1158	553	1939	124

* KNICS (Korea Nuclear Instrumentation and Control System Research and Development Center)
** Doosan Heavy Industry & Construction, KNICS-RPS-SDS231-01, Rev. 01, Software Design Specification for the Bistable Processor of the Reactor Protection System (2006)

Test Suite Generation (1/2)

- Step2-1: Generate a test suite.
  - Generate test suites achieving **100% feasible coverage** for each of the three test criteria and five subject models.

Input

Test criterion ***C***

Subject model **M**

FBDTester

Automatic test suite generation tool for FBD models

Output

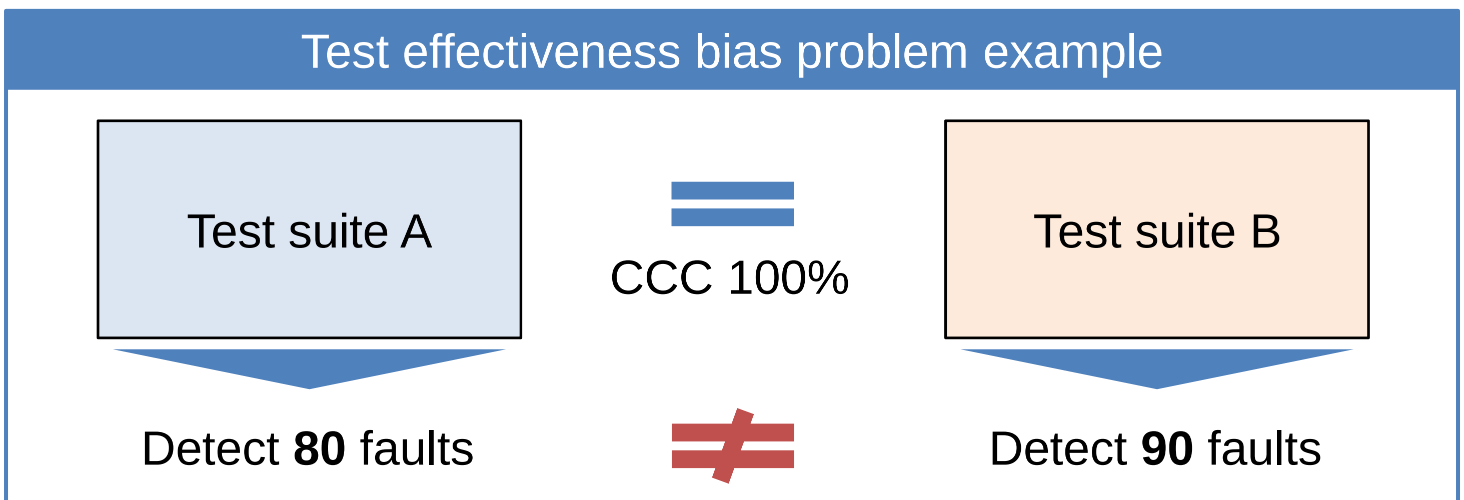
Test suite $S_{C,M}$

(**Set of test cases** for **M** satisfying **C** maximally)

* Eunkyoung Jee: A Data Flow-based Structural Testing Technique for FBD Programs. Ph.D Thesis. KAIST press, Republic of Korea (2009)

Test Suite Generation (2/2)

- Step2-2: Generate 100 test suites.
  - Repeat “step2-1” 100 times to generate 100 test suites for each test criterion and FBD model.
    - ✓ Total 3 × 5 × 100 = **1,500 test suites**
  - Because of the test effectiveness **bias problem**
    - ✓ **Two test suites** satisfying the same test coverage criteria may **differ widely in their fault detection** effectiveness.



Mutant Generation (1/2)

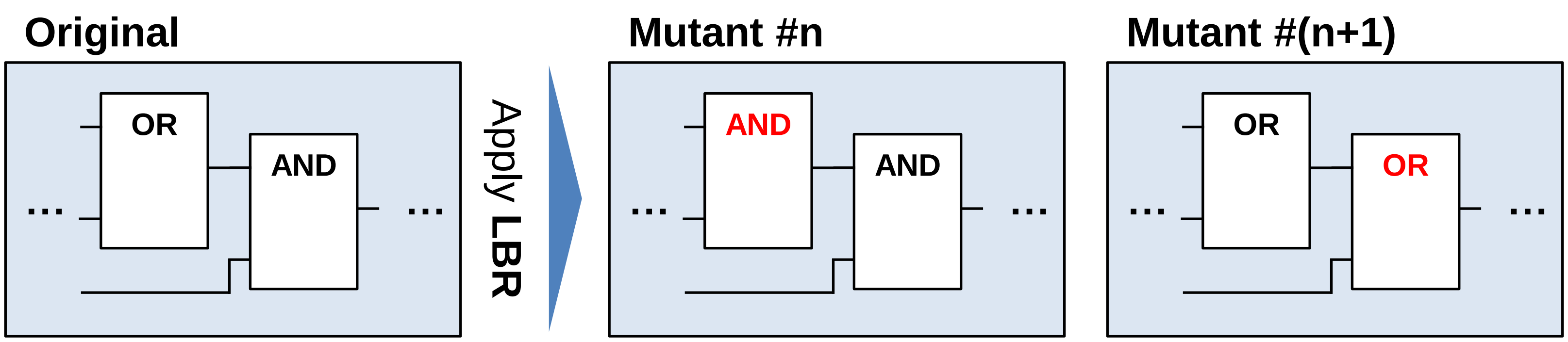
- Step3-1: Define FBD mutation operators
  - Widely survey industry FBD faults.
    - ✓ Fundamental principle: mutants could represent **the mistakes that programmers often make**.
  - Refer existing good practice for using mutation operators.
    - ✓ Selective mutation: there are five **key mutation operators** achieved 99.5% fault detection ability [Offutt et al.]

FBD mutation operator	Description
CVR (Constant Value Replacement)	Replace a integer constant C by C-2, C-1, C+1, C+2
IID (Inverter Insertion or Deletion)	Negate a Boolean edge
ABR (Arithmetic Block Replacement)	Replace an arithmetic block from the same class (ADD, SUB, DIV, MUL, EXP, MOD)
CBR (Comparison Block Replacement)	Replace a comparison block from the same class (LT, LE, GT, GE, EQ, NE)
LBR (Logic Block Replacement)	Replace a logical block from the same class (OR, AND, XOR)

Mutant Generation (2/2)

- Step3-2: Generate FBD mutants.
  - Apply the FBD mutation operators to each block or edge of the subject models whenever possible.

Example: Applying FBD mutation operators



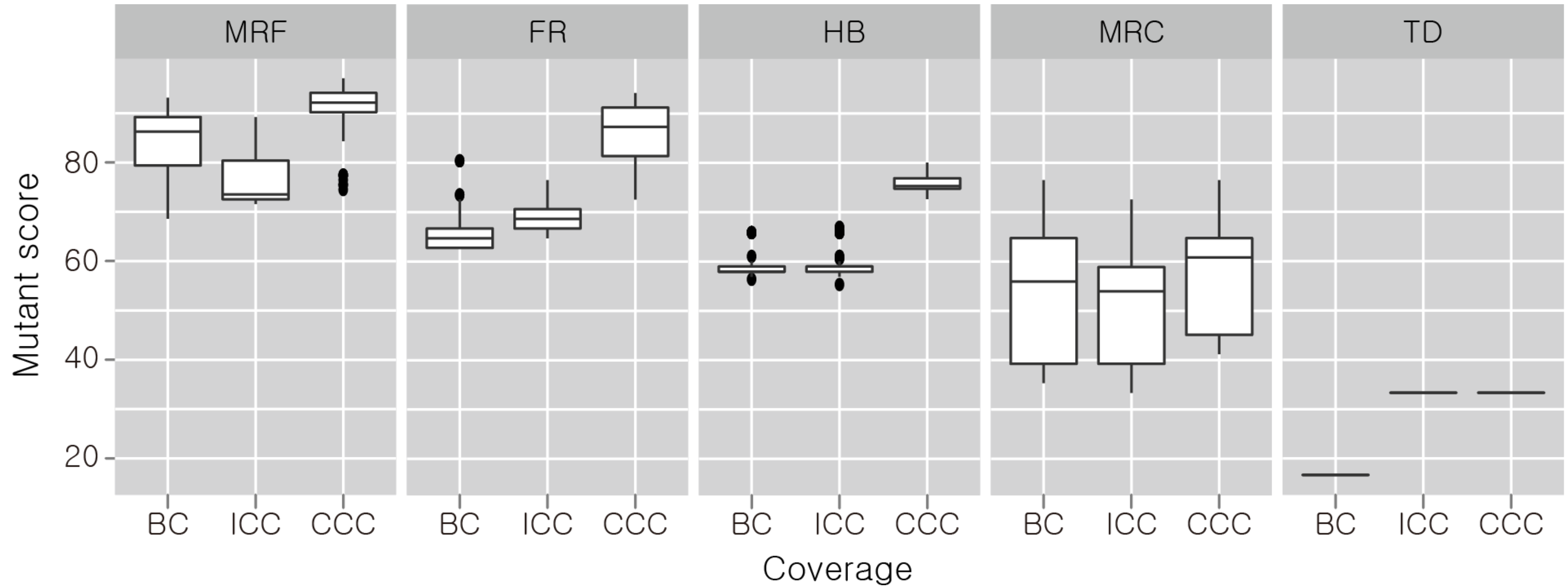
Results

Subject FBD model	FR	HB	MRC	MRF	TD	Total
Number of mutants	102	190	51	102	36	481

Outline

- Approach Overview
  - Research questions
  - Mutation analysis
  - Study method
- Experimental Description
  - Subject models
  - Test suite generation
  - Mutant generation
- Analysis Results
- Summary and Future Work

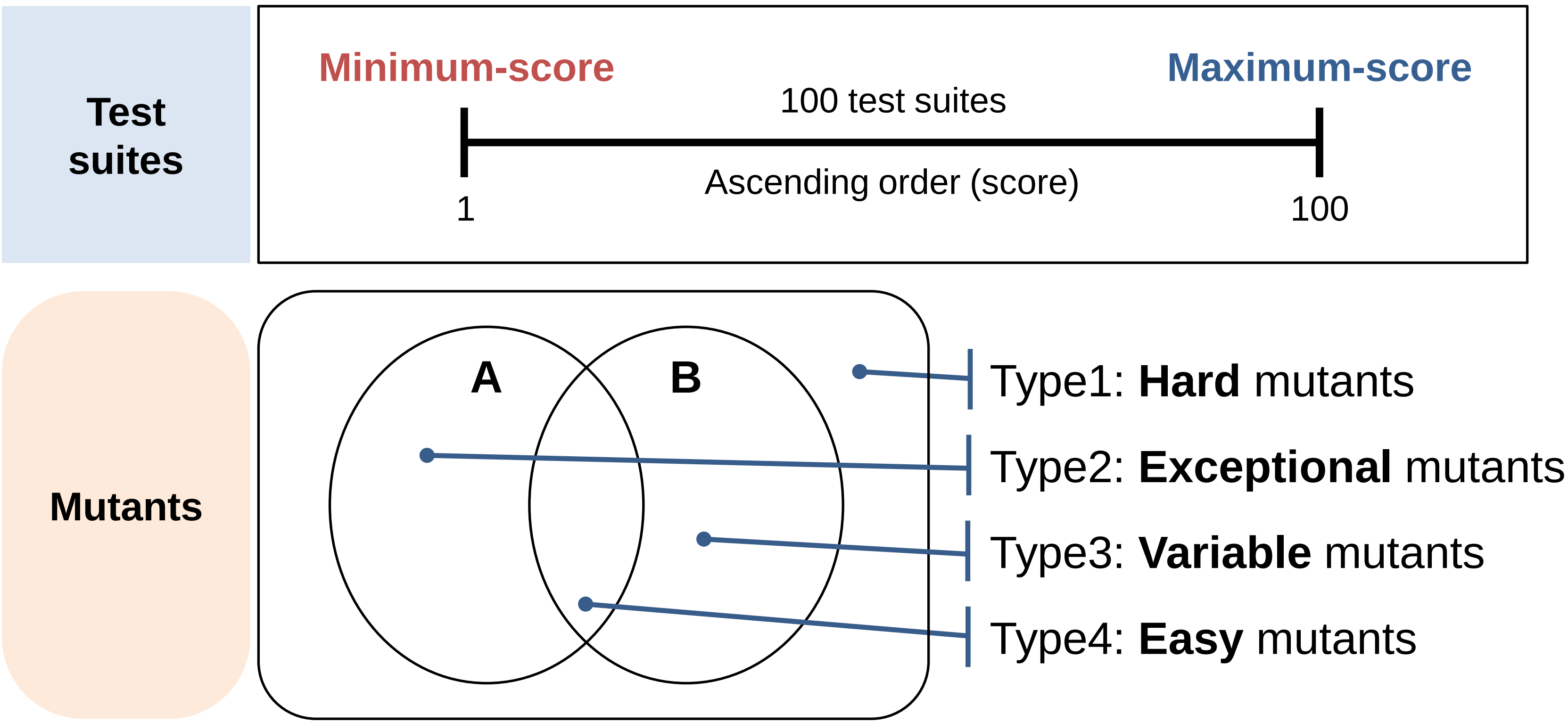
Q1: Fault Detection Effectiveness



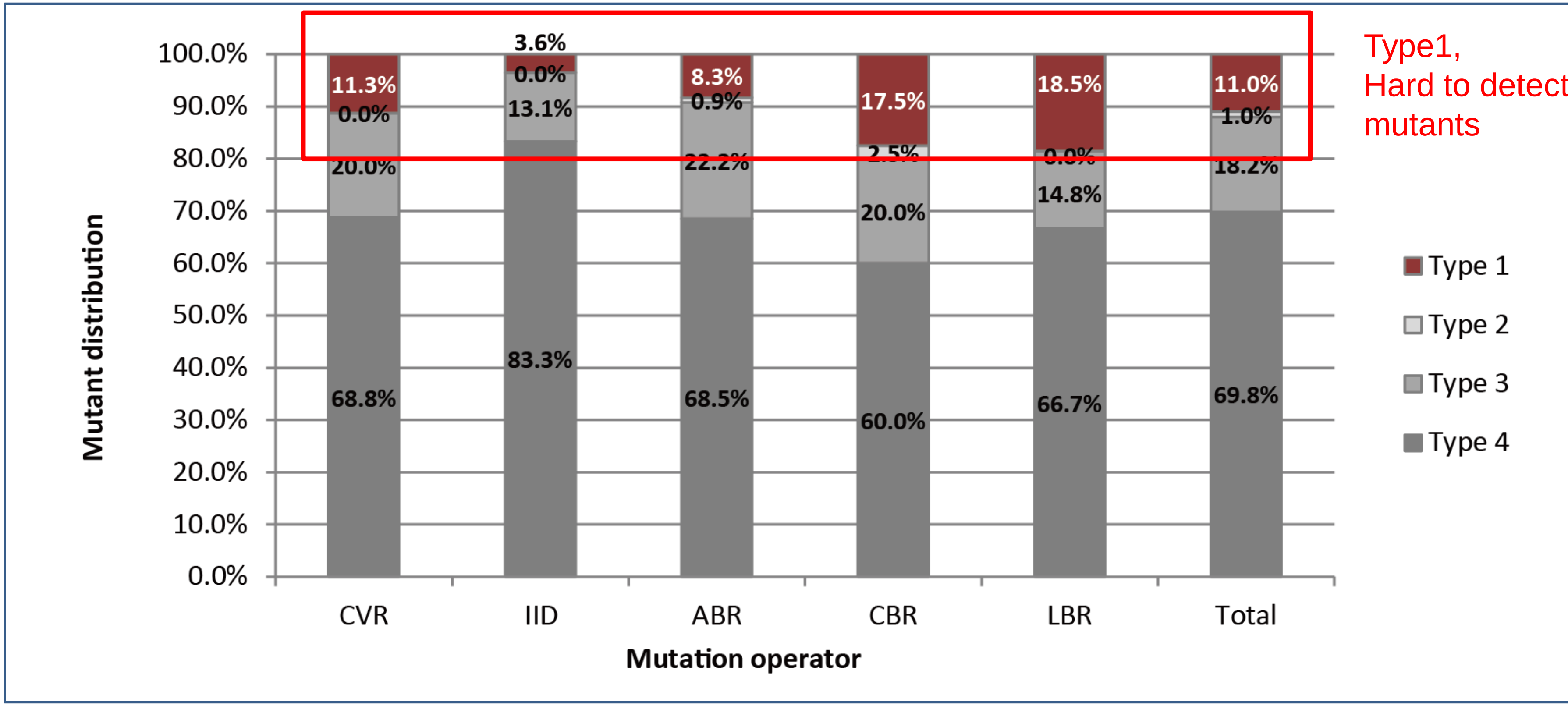
1	Statistically, **BC(64.7) < ICC(68.6) < CCC(81.2)** in the **average** mutant score except TD.
2	The fault detection effectiveness of CCC is **up to 97.1%**.
3	The fault detection ability **varies widely** depending on the FBD models. This is related with Q2.

Q2: Strength and Weakness (1/2)

- Focus on fault detection probability
  - Mutants detected by the minimum-score suite → **Set A**
  - Mutants detected by the maximum-score suite → **Set B**



Q2: Strength and Weakness (2/2)



1	FBD test criteria have **strength** on detecting faults in Boolean edges (**IID**) and arithmetic blocks (**ABR**).
2	FBD test criteria have **weakness** on detecting faults in comparison blocks (**CBR**), logic blocks (**LBR**), and constants (**CVR**).

Outline

- Approach Overview
  - Research questions
  - Mutation analysis
  - Study method
- Experimental Description
  - Subject models
  - Test suite generation
  - Mutant generation
- Analysis Results
- Summary and Future Work

Summary

- Key findings
  - **Test effectiveness** of existing FBD test criteria
    - ✓ In average, [**BC=64.7**], [**ICC=68.6**], and [**CCC=81.2**].
  - Fault detection strength and weakness
    - ✓ **Strong**: faults in **Boolean** edges and **arithmetic** blocks.
    - ✓ **Weak**: faults in **comparison**, **logic** blocks and **constants**.
- Future work
  - Investigate more FBD mutation operators.
  - Compare to other test criteria used in code-level.

Thank you ☺

Donghwan Shin @ KAIST SE Lab.

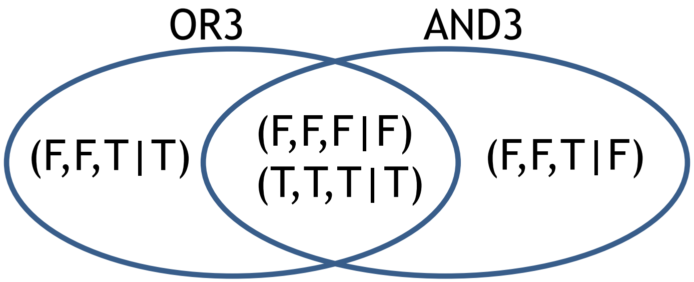
donghwan@se.kaist.ac.kr

Appendix

- One root cause of weakness
- Details for tools
- FBDTester (inside)

Appendix

- One root cause of weakness
  - BC, ICC and CCC is focused on Boolean inputs and outputs.
  - **Do not consider input combination.**
    - ✓ ex) OR3(a, b, c | d) → CCC-suite 100%: (F, F, F | F) and (T, T, T | T) and this CCC-suite is exactly **same** as AND3 block.

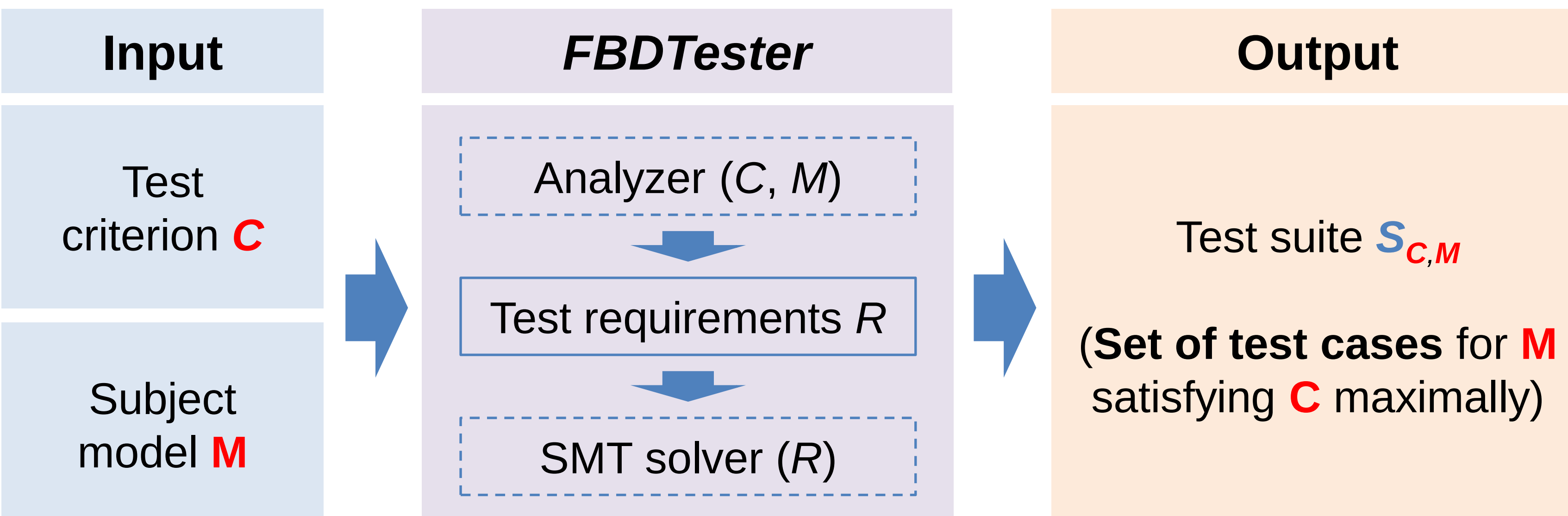


Appendix

- Tools
  - *MutantGenerator*: mutant generation tool
    - ✓ Applying the given mutation operators, *MutantGenerator* automatically generate all available mutants for a target FBD model.
    - ✓ All equivalent mutants are removed by manual inspection.
    - ✓ Input: FBD model (XML format)
    - ✓ Output: mutants for the model (XML format), mutants information (txt format)
  - *MutantSimulator*: mutant simulation tool
    - ✓ From a clean FBD model and mutants, *MutantSimulator* automatically run and compare results to count the kill score.
    - ✓ Input: FBD model (XML, clean version), plenty of mutants (XML format)
    - ✓ Output: mutant score, killed information (txt format)

Appendix

- FBDTester
  - FBDTester uses **SMT solver** engine to solve the set of **test requirement assertions**, and the solution of this SAT problem is a set of test cases for the test requirements.



* Eunkyoung Jee: A Data Flow-based Structural Testing Technique for FBD Programs. Ph.D Thesis. KAIST press, Republic of Korea (2009)