An Empirical Comparison of Compiler Testing Techniques

报告人介绍 – 熊英飞

- 2000~2004，电子科技大学本科
- 2004~2006，北京大学研究生
  - 导师：梅宏、杨芙清
- 2006~2009，日本东京大学博士
  - 导师：胡振江、武市正人
- 2009~2011，加拿大滑铁卢大学博士后
  - 导师：Krzysztof Czarnecki
- 2012~，北京大学“百人计划”研究员
- 研究方向：软件分析、编程语言设计

北京大学软件工程研究所

- 国内最早开展软件工程研究、规模最大、最有影响力的软件工程研究团队
- 院士三名（含双聘一名），博士生导师10名，硕士生导师13名
- 在软件工程顶级会议发表论文数为大陆第一名
- 发表了中国大陆第一篇ICSE，第一篇FSE，第一个CCF A类会议上的杰出/最佳论文奖

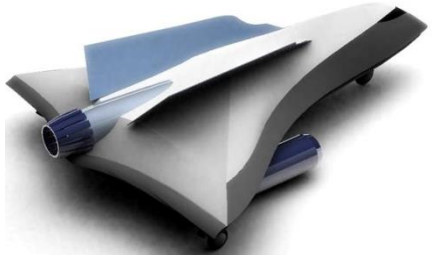
Compilers are important!

Compiler

Operating System

App

……



Safe-critical System

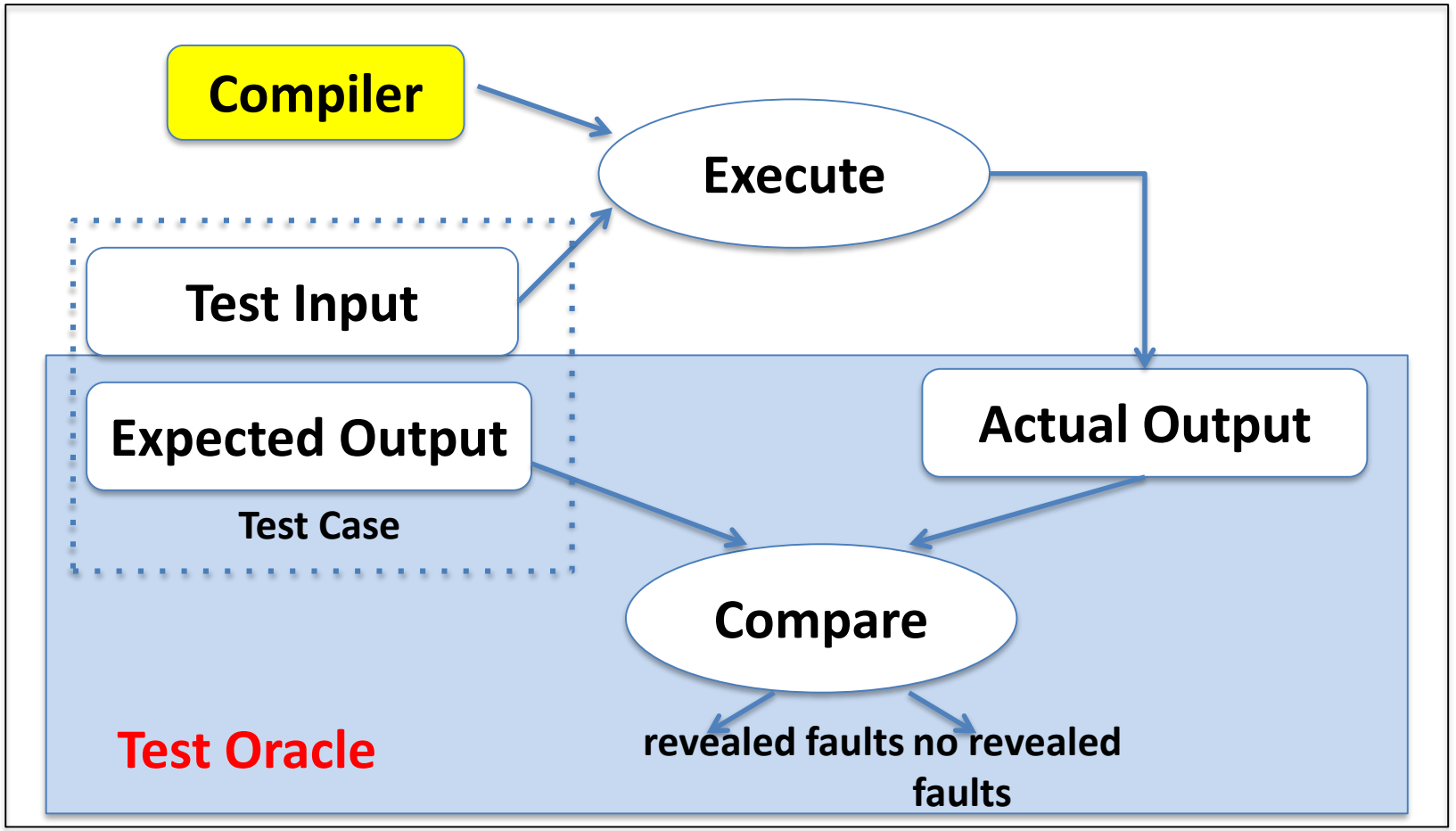
Buggy Compilers

Harmful!!!

- **Unintended behaviors of developed programs**
- **Increased debugging difficulty**
- **……**

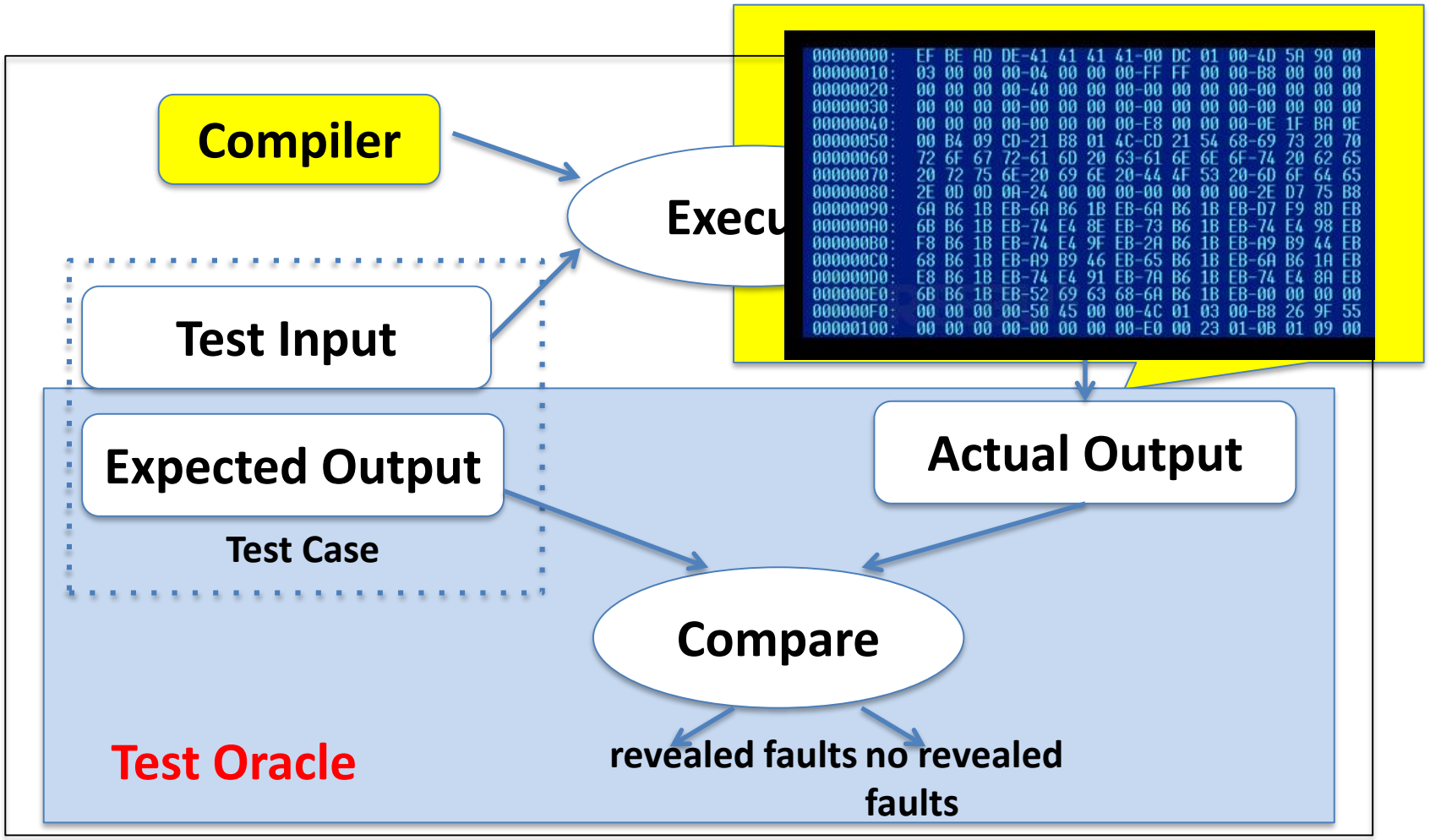
Compiler Testing

--- guaranteeing compiler quality



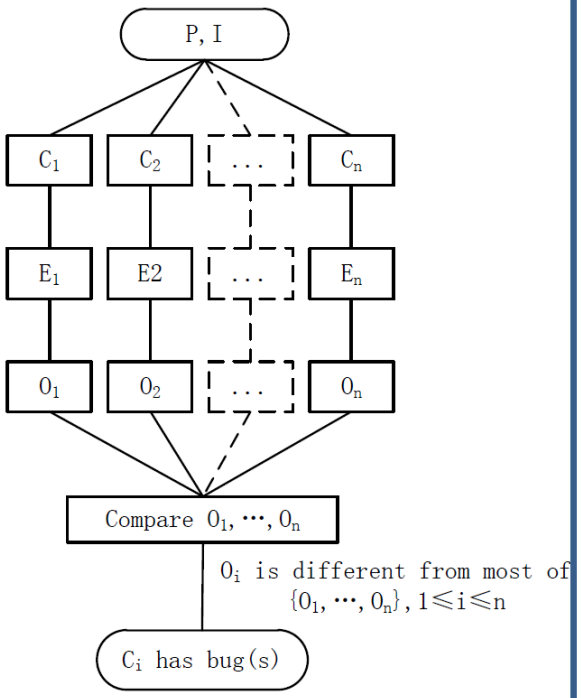
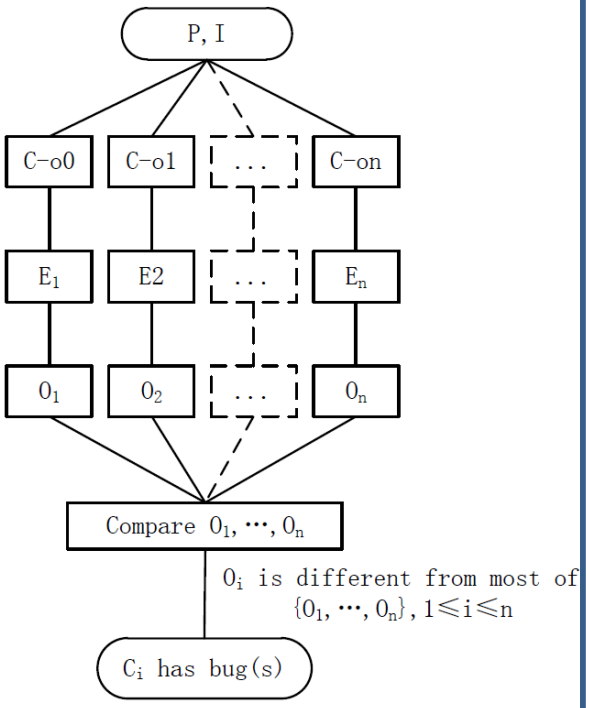
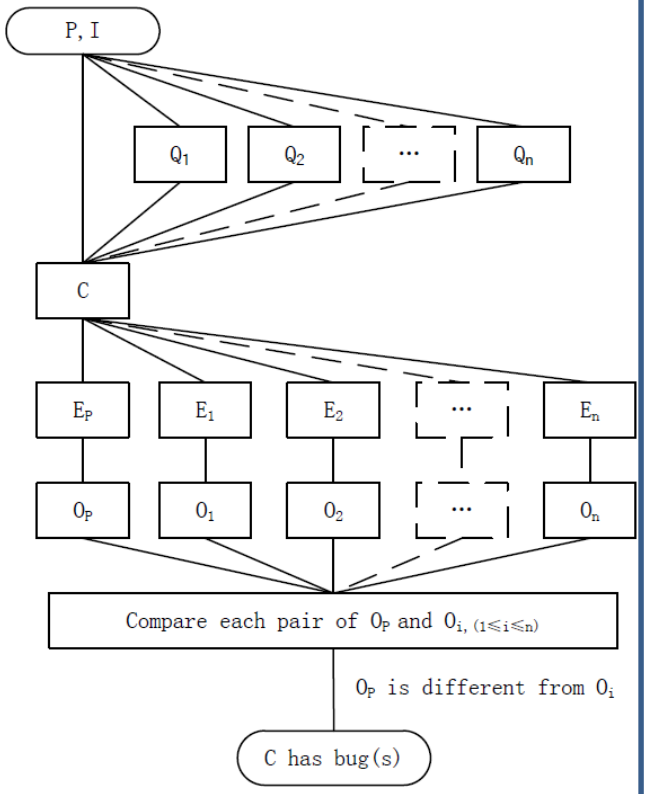
Software Testing Process

Test Oracle
---One Challenge in Compiler Testing



Software Testing Process

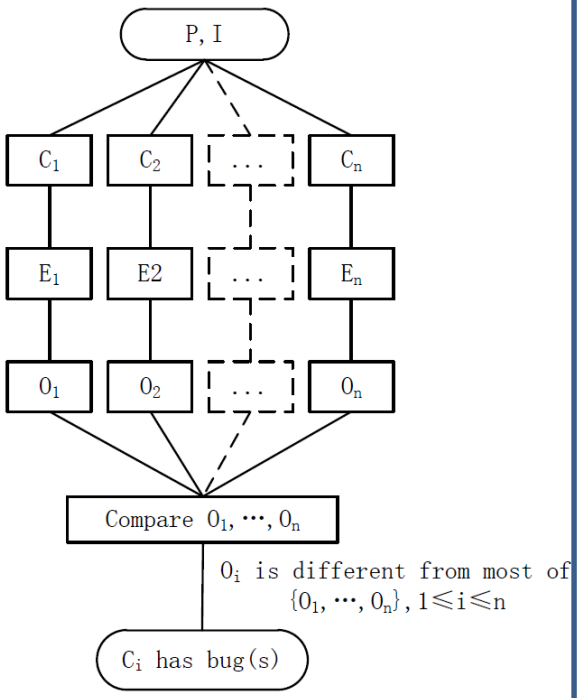
Compiler Testing Techniques

RDT[1]	DOL	EMI[2]
		

[1] W. M. McKeeman. Differential testing for software. Digital Technical Journal, 1998.
[2] V. Le, M. Afshari, Z. Su. Compiler validation via equivalence modulo inputs, PLDI, 2014.

Randomized Differential Testing

RDT[1]

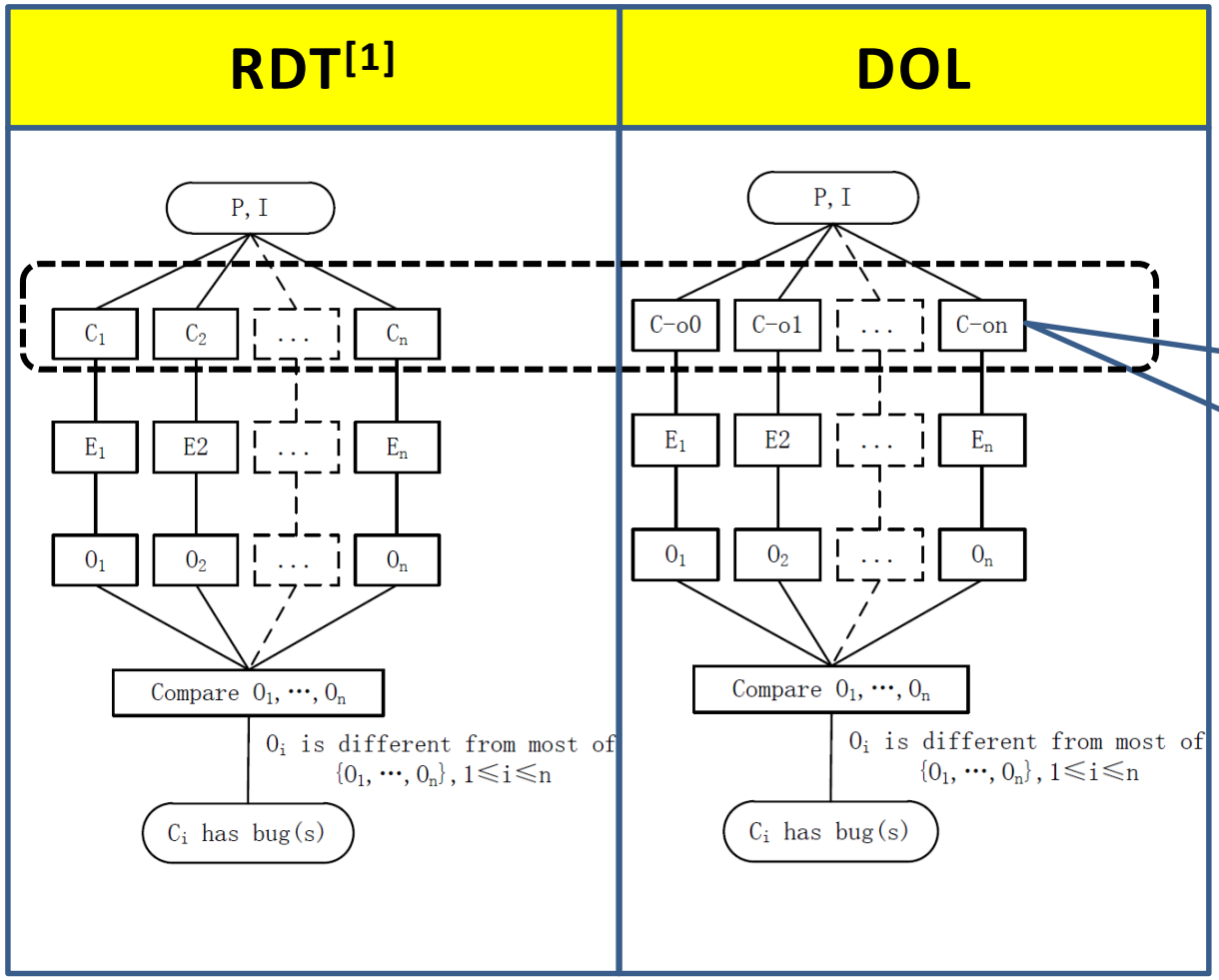


- Assume different compilers are implemented based on the same specification
- Detect bugs by comparing the outputs of these compilers for the same test program

Test Oracle:
≥2 **comparable compilers**

[1] W. M. McKeeman. Differential testing for software. Digital Technical Journal, 1998.

Different Optimization Levels --- variant of RDT



Compare the output of **one compiler at different optimization levels** for the same test program

Test Oracle:
A compiler with various optimization levels

[1] W. M. McKeeman. Differential testing for software. Digital Technical Journal, 1998.

Equivalence Modulo Inputs

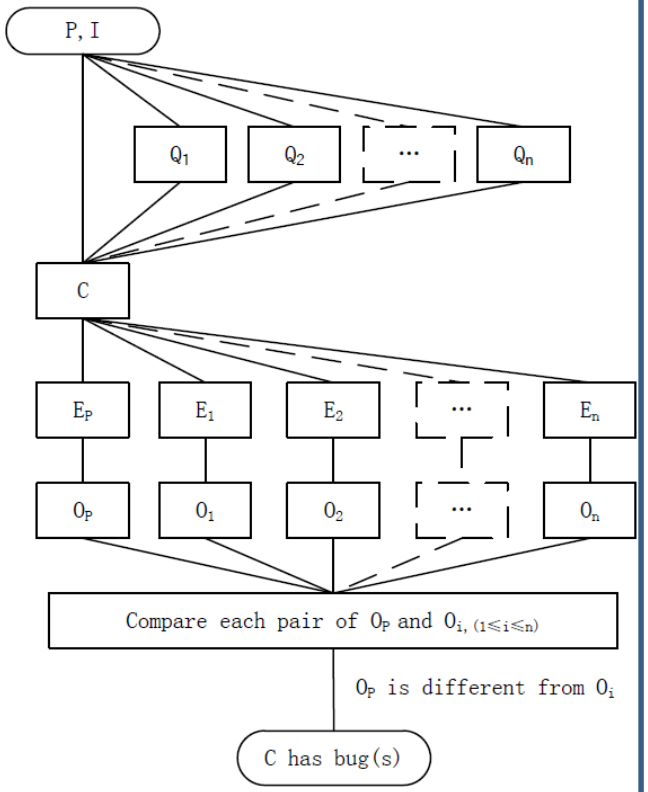
- Comparison between a test program and its variants whose behaviors are regarded as equivalent under a set of test inputs for this test program

```
main(){
  int x=1;
  if(x==1)
    printf("x=1");
  if(x==2)
    printf("x=2");
  …
}
```
P

```
main(){
  int x=1;
  if(x==1)
    printf("x=1");
  if(x==2)
    printf("x=2");
  …
}
```
Q_i

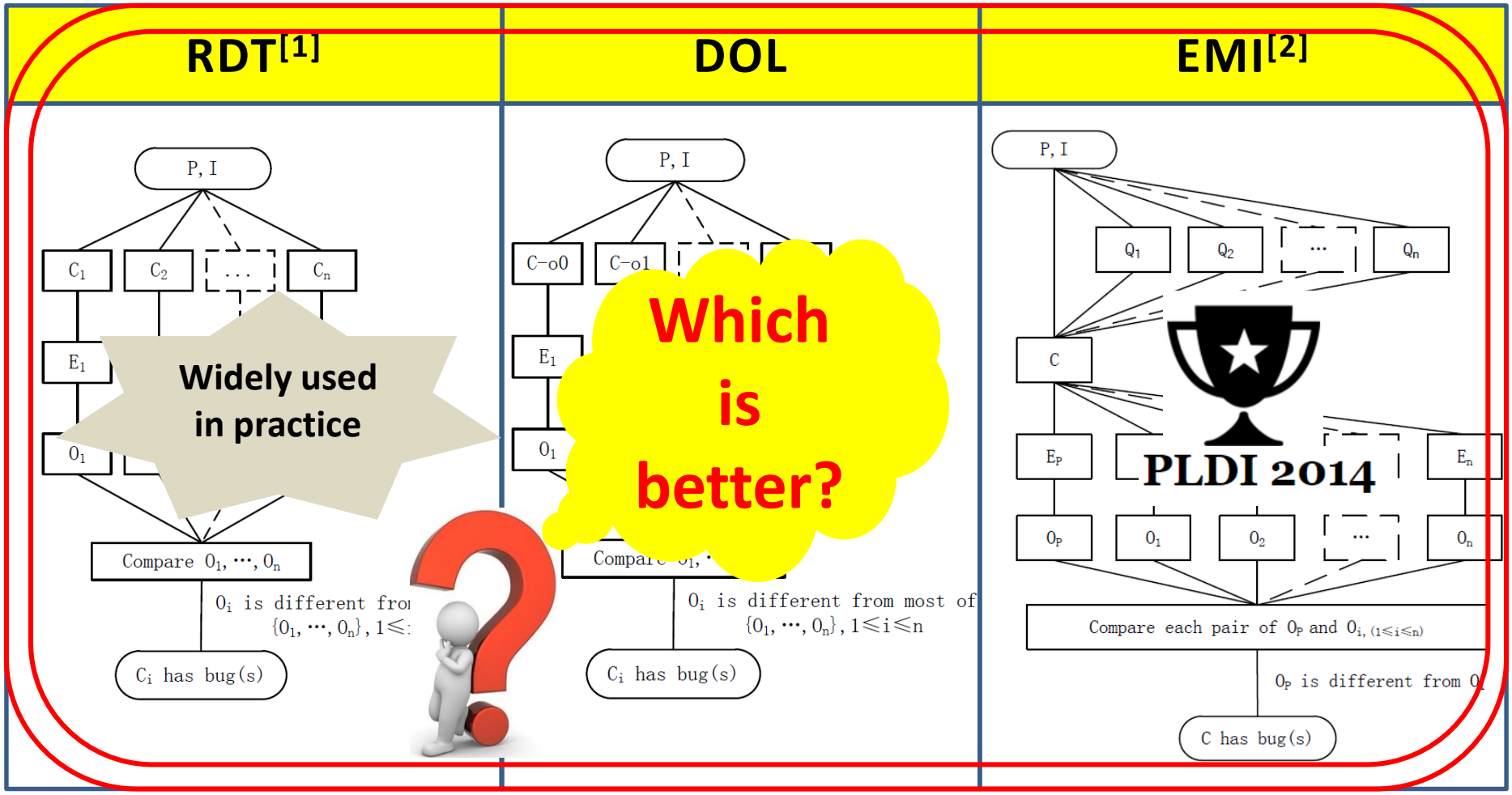
Test Oracle:
Program and its variants with equivalent behaviors under some test inputs

EMI[2]



[2] V. Le, M. Afshari, Z. Su. Compiler validation via equivalence modulo inputs, PLDI, 2014.

Systematic and Comprehensive Empirical Comparison



[1] W. M. McKeeman. Differential testing for software. Digital Technical Journal, 1998.

[2] V. Le, M. Afshari, Z. Su. Compiler validation via equivalence modulo inputs, PLDI, 2014.

Measurement

Ideal:
number of detected bugs

Two Alternative Measurements

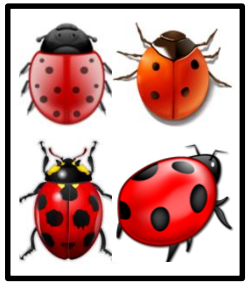
- **Number of bugs manually identified**

  Scalability Problem

- **Number of test programs triggering bugs**

  Highly Inaccurate

A New Measurement



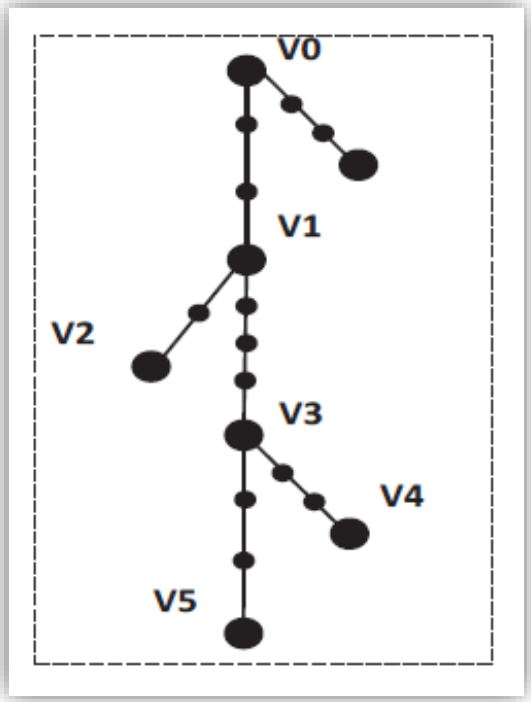
Correcting Commits

For any test program triggering a bug of a compiler C whose commit version is x (e.g., V0)

- check subsequent commits of the compiler and determine which commit corrects the bug.

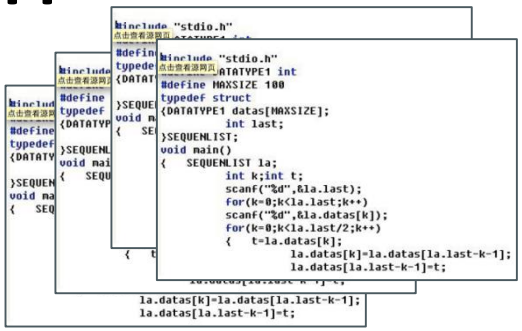
Same Bug:

- **the version triggering the bug--- depth-first algorithm**
- **the version correcting the bug--- binary search algorithm**



Study Design

Csmith



RDT[1]	DOL	EMI[2]

GCC 4.4.3

LLVM 2.6 (Clang)

90 hours

Four Studies

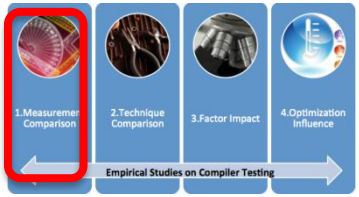
1.Measurement Comparison

2.Technique Comparison

3.Factor Impact

4.Optimization Influence

Empirical Studies on Compiler Testing

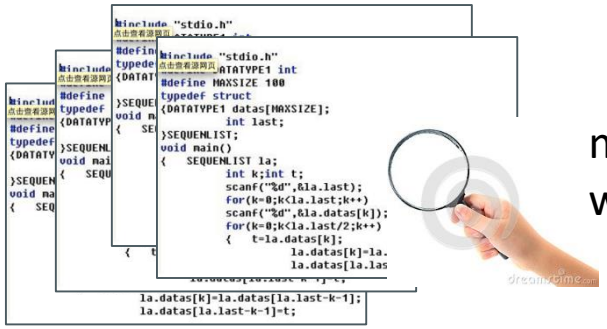


Study 1: Measurement Comparison

accuracy of the number of test programs

Table 1: Number of Test Programs Triggering Bugs

Bug	1st	2nd	3rd	4th	5th
Test Programs Triggering Bugs	17	1	1	1	877



manually check five commits of GCC, each of which fixes only one GCC bug

Finding 1: (on measurement)

- **Number of test programs triggering bugs is inaccurate**
- **Number of correcting commits is necessary**

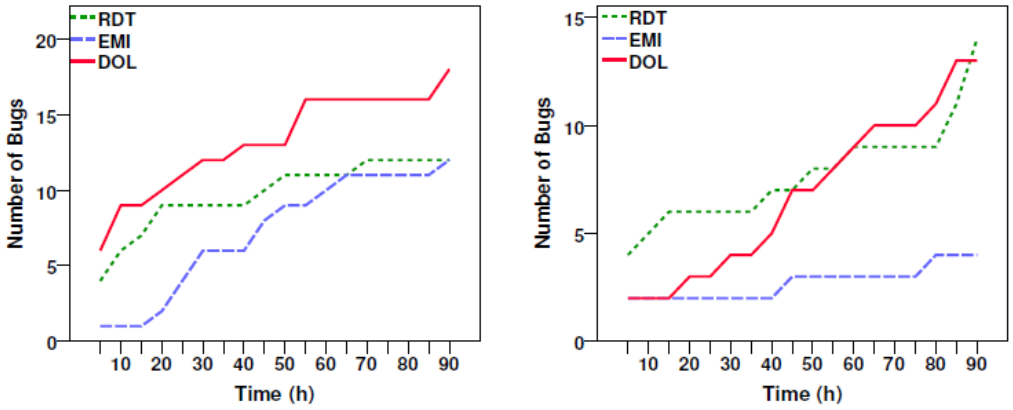
Study 2: Effectiveness Comparison

number of detected bugs

Table 2: Effectiveness Comparison

Compilers	Detected Bugs			Test Programs that Trigger Bugs		
	RDT	EMI	DOL	RDT	EMI	DOL
GCC	12	12	18	422	492	954
LLVM	14	4	13	801	6	54
TOTAL	26	16	31	1,223	498	1,008

number of bugs detected per 10 hours



(a) GCC (b) LLVM

time to detect the first bug

Table 3: Time to Detect the First Bug (seconds)

Compilers	RDT	EMI	DOL
GCC	116	978	84
LLVM	34	17,571	11,363

Study 2: Effectiveness Comparison

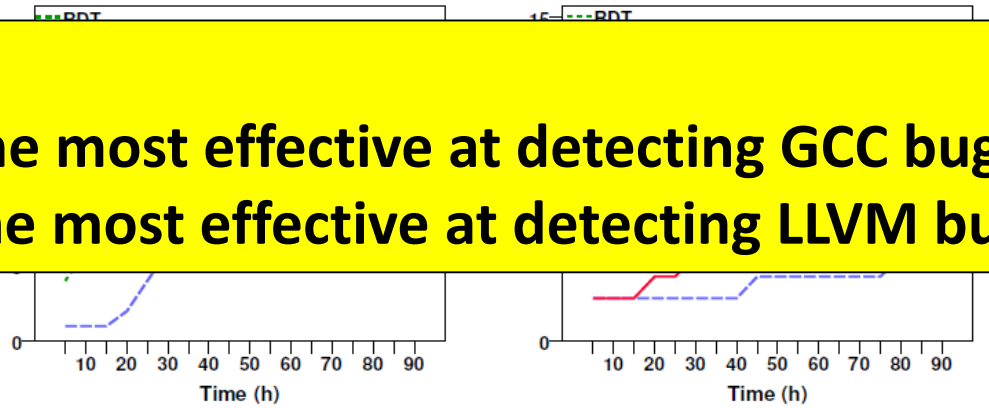
number of detected bugs

Table 2: Effectiveness Comparison

Compilers	Detected Bugs			Test Programs that Trigger Bugs		
	RDT	EMI	DOL	RDT	EMI	DOL
GCC	12	12	18	422	492	954
LLVM	14	4	13	801	6	54
TOTAL	26	16	31	1,223	498	1,008

Finding 2:

- **DOL seems to be the most effective at detecting GCC bugs**
- **RDT seems to be the most effective at detecting LLVM bugs**



(a) GCC (b) LLVM

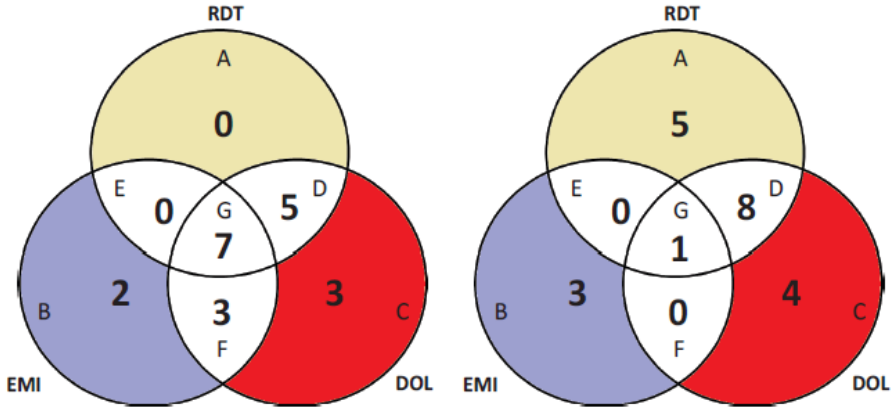
time to detect the first bug

Table 3: Time to Detect the First Bug (seconds)

Compilers	RDT	EMI	DOL
GCC	116	978	84
LLVM	34	17,571	11,363

Study 2: Substitutability Comparison

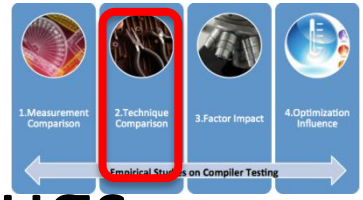
- **Unique Bugs: bugs detected by only one compiler testing technique**
- **More unique bugs ➔ less substitutable**



(a) GCC (b) LLVM

Finding 3:

- **DOL: more effective at detecting GCC unique bugs**
- **RDT: more effective at detecting LLVM unique bugs**
- **RDT can be substituted by DOL completely in detecting GCC bugs**



Study 2: Optimization-Related/Irrelevant Bugs

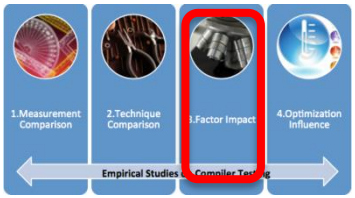
- **optimization-related bugs: bugs detected through DOL**
- **optimization-irrelevant bugs: otherwise**

Table 4: Comparison on Optimization-Related Bugs and Optimization-Irrelevant Bugs

Bugs	Optimization-Related			Optimization-Irrelevant		
	RDT	EMI	DOL	RDT	EMI	DOL
GCC	12	12	18	0	0	0
LLVM	9	4	13	5	0	0
TOTAL	21	16	31	5	0	0

Finding 4:

- **Optimization-related bugs: DOL seems to be more effective**
- **Optimization-irrelevant bugs: RDT seems to be more effective**
- **GCC may have more optimization-related bugs**



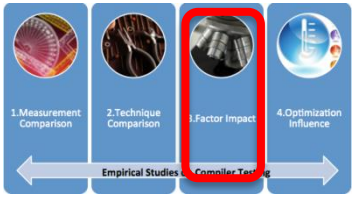
Study 3: Impact of Efficiency

- **Efficiency: #test programs being tested given a fixed period of time**

Table 5: Number of Test Programs Used During the Experimental Period

Compilers	RDT	EMI	DOL
GCC	27,990	4,794	62,552
LLVM	27,990	5,040	64,385

Finding 5:
DOL is the most efficient, whereas EMI is the least efficient one



Study 3: Strength of test oracles

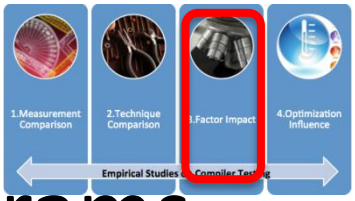
- **Strength of test oracles: given the same test programs, which technique detects more bugs?**

Table 6: Strength of Test Oracles

Compilers	Detected Bugs			Unique Bugs		
	RDT	EMI	DOL	RDT	EMI	DOL
GCC	18	12	18	0	0	0
LLVM	16	4	10	6	0	0
Total	34	16	28	6	0	0

Finding 6:

- **RDT and DOL oracles are stronger than EMI oracle.**
- **RDT oracle is not weaker than DOL oracle.**



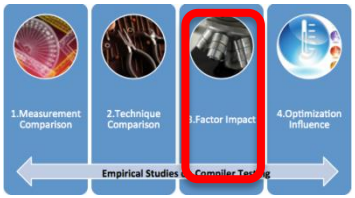
Study 3: Effectiveness of generated test programs

- **Effectiveness of generated test programs:**
  - **randomly generated programs**
  - **variants generated by EMI**

Table 7: Effectiveness of Generated Test Programs

Compilers	RDT		DOL	
	Random	Variant	Random	Variant
GCC Bugs	11	8	11	8
LLVM Bugs	9	6	4	2
GCC Unique Bugs	6	3	6	3
LLVM Unique Bugs	5	2	3	1

Finding 7:
Variants generated by EMI are less effective than randomly generated programs by CSmith, but they are a complement to randomly generated programs.



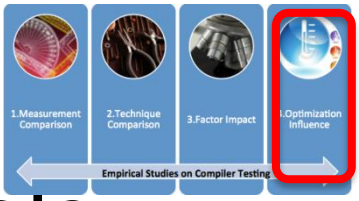
Study 3: Statistical Analysis

- **Efficiency**
- **Strength of test oracles**
- **Effectiveness of generated test programs**

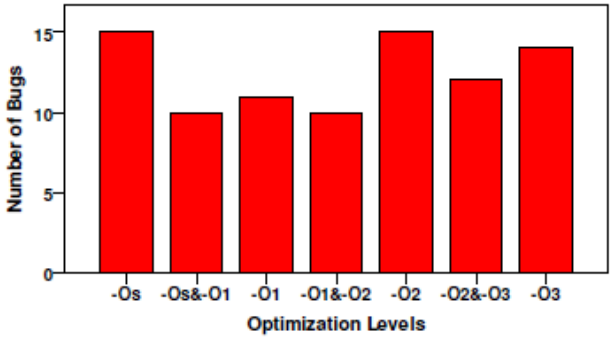
Factors	Sig
Test Oracle (RDT v.s. DOL)	0.041
Program (Random v.s. Variant)	0
Efficiency	0
Test Oracle (EMI v.s. RDT)	0
Test Oracle (EMI v.s. DOL)	0.002
Efficiency	0

Finding 8:

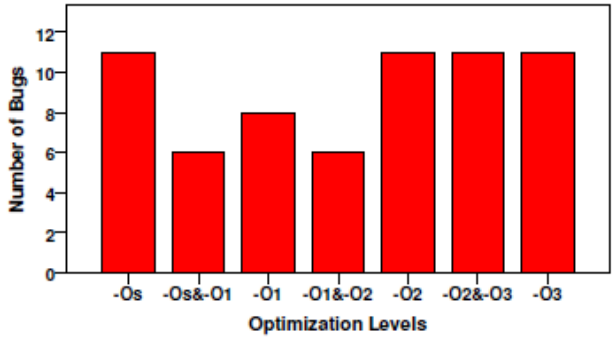
- **All the three factors have statistically significant impact**
- **Efficiency has the most significant impact**
- **The effectiveness of generated test programs has the least significant impact.**



Study 4: Influence of Optimization Levels



(a) GCC

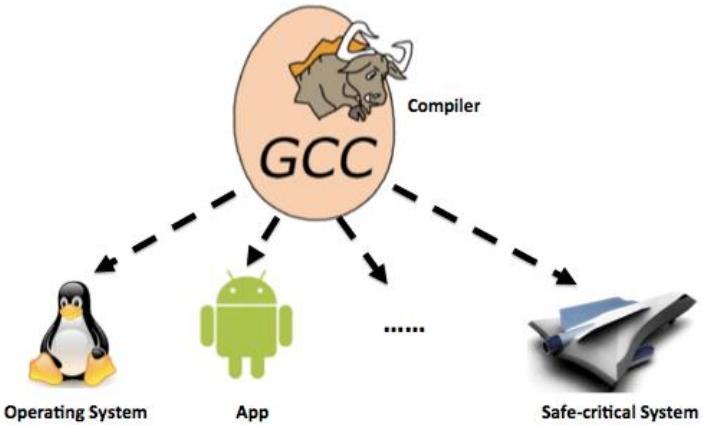


(b) LLVM

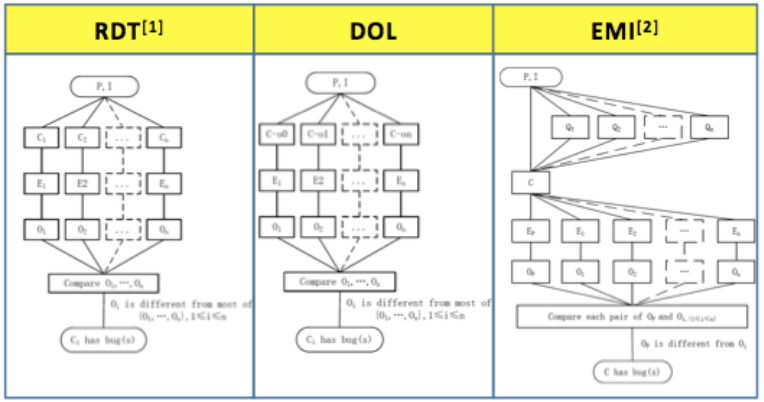
Finding 9:
As some compiler bugs are only triggered by lower optimization levels alone, it is necessary to test a compiler with various optimization levels.

Conclusions

Compilers are important!



Compiler Testing Techniques



[1] W. M. McKeeman. Differential testing for software. Digital Technical Journal, 1998.
[2] V. Le, M. Afshari, Z. Su. Compiler validation via equivalence modulo inputs, PLDI, 2014.

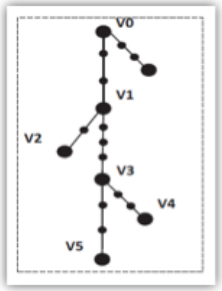
Correcting Commits

For any test program triggering a bug of a compiler C whose commit version is x (e.g., V0)

- check subsequent commits of the compiler and determine which commit corrects the bug.

Same Bug:

- **the version triggering the bug--- depth-first algorithm**
- **the version correcting the bug--- binary search algorithm**



Four Studies

Thank You