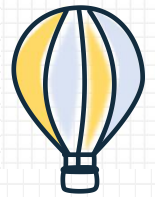
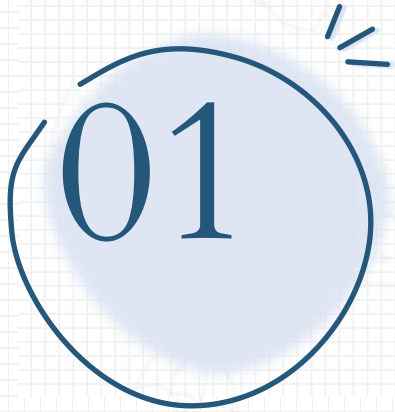




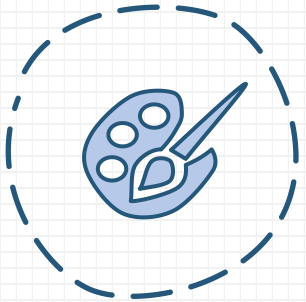
从变异测试到编译器测试

汇报人：陈冰婷

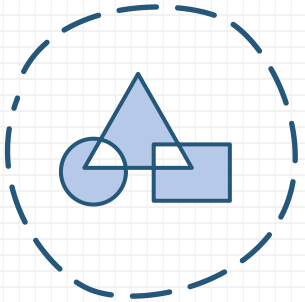


What Is Mutation Testing

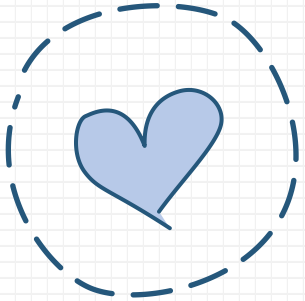
Mutation Testing



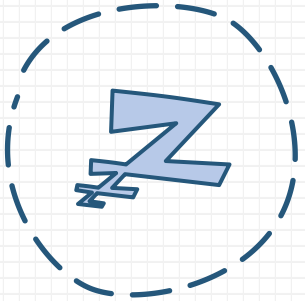
是一种基于故障的测试技术



可以追溯到20世纪70年代



提供了一种强测试标准



工业上使用该标准较少

Mutation Testing

```
begin
    int x,y;
    input(x,y);
    if(x!=y)
      output(x+y);
    else
      output(x*y);
end
```

$$T_1 = \{(x = 1, y = 1), (x = 1, y = 2)\}$$

$$MS(T) = \frac{|D|}{|M| - |E|}$$

1) $|D|$ = #killed mutants
2) $|M|$ = #mutants
3) $|E|$ = #equivalent mutants

```
begin
    int x,y;
    input(x,y);
    if(x<y)
      output(x+y);
    else
      output(x*y);
end
```

>

Mutant

Mutation Testing

This table lists the first set of formalized mutation operators for the Fortran programming language.

These typical mutation operators were implemented in the Mothra mutation system

序号	变异算子	描述
1	AAR	用一数组引用替代另一数组引用
2	ABS	插入绝对值符号
3	ACR	用数组引用替代常量
4	AOR	算术运算符替代
5	ASR	用数组引用替代变量
6	CAR	用常量替代数组引用
7	CNR	数组名替代
8	CRP	常量替代
9	CSR	用常量替代变量
10	DER	DO语句修改
11	DSA	DATA语句修改
12	GLR	GOTO标签替代
13	LCR	逻辑运算符替代
14	ROR	关系运算符替代
15	RSR	RETURN语句替代
16	SAN	语句分析
17	SAR	用变量替代数组引用
18	SCR	用变量替代常量
19	SDL	语句删除
20	SRC	源常量替代
21	SVR	变量替代
22	UOI	插入一元操作符

02

How Does It Work

Mutation Testing

p	被测程序
T	设定的一个测试用例集合
p'	变异算子

survived	可存活变异体
killed	可杀除变异体
Equivalent Mutant	等价变异体

$$MS(T) = \frac{|D|}{|M| - |E|}$$

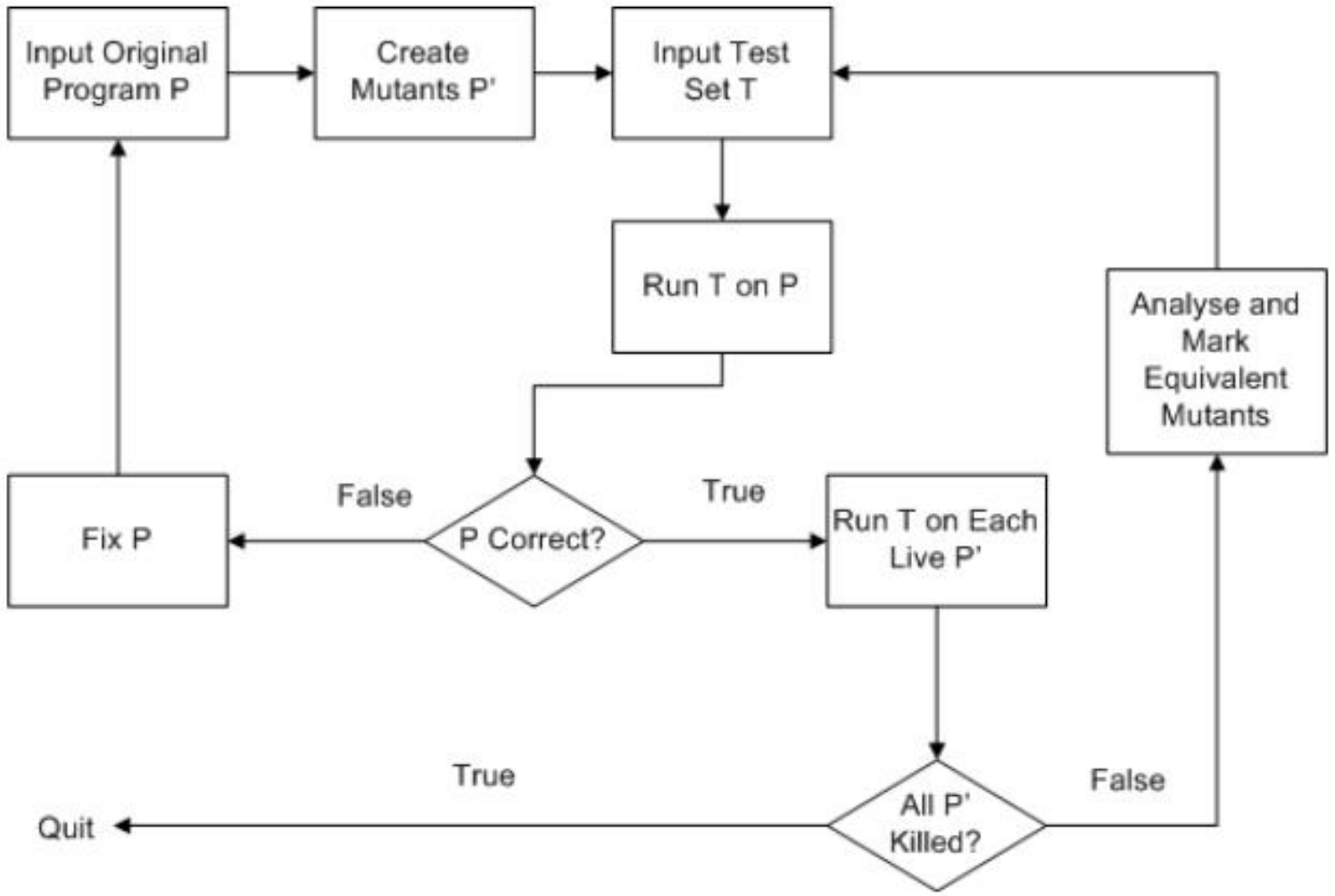
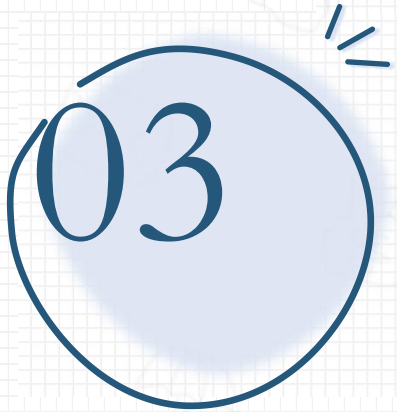
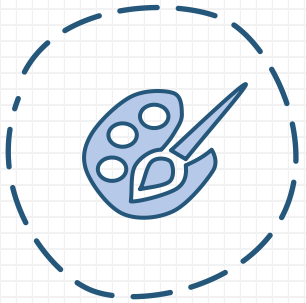


Fig. 2. Generic Process of Mutation Analysis [191]

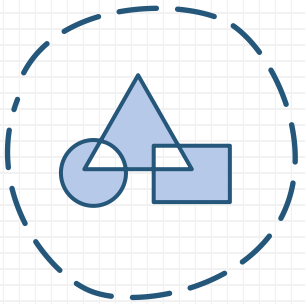


Why Use Compiler Testing

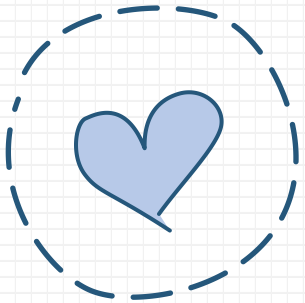
Compiler Testing



是对于编译器的测试

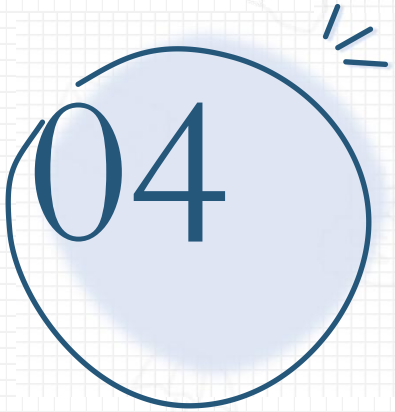


编译器错误会导致从正确的源代码生成不正确的二进制代码



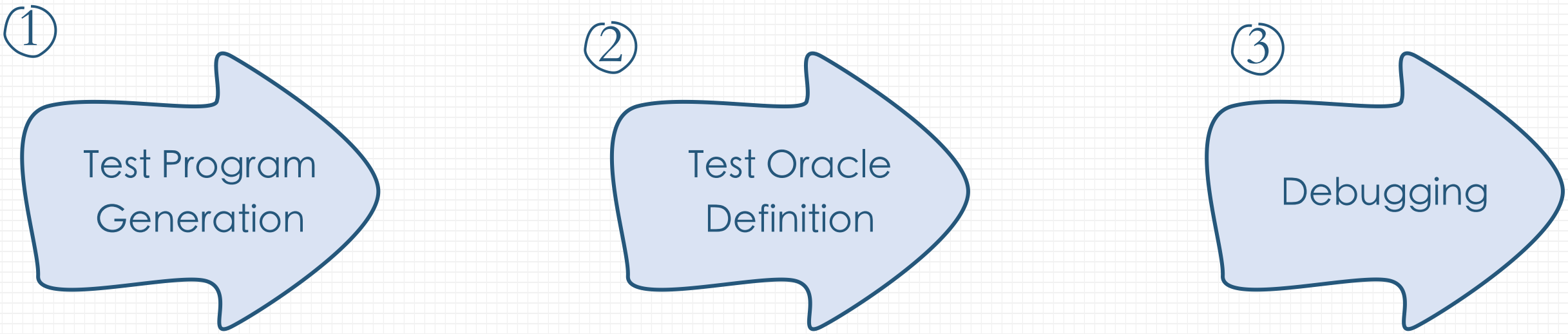
大多数基于程序变异的编译器测试方法

大多数研究工作是由Csmith的成功所推动的



From Mutation Testing to Compiler Testing

Compiler Testing



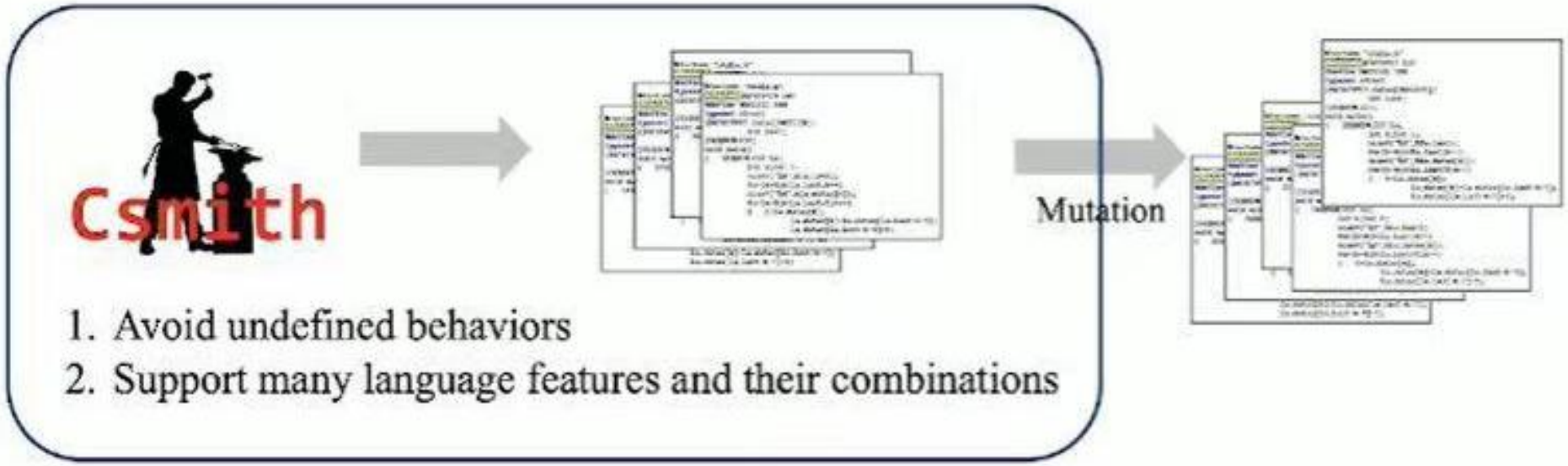
Constructing Test Programs

Constructing Test Programs	
Manually Constructing Test Programs	
Test Program Generation	Grammar-directed Approaches
	Grammar-aided Approaches
	Other Approaches
Program Mutation	Semantics-Preserving Mutation
	Non-Semantics-Preserving Mutation

Table 1. Overview of approaches for constructing test programs.

Test Program Generation Through Mutation

- Construction v.s. Mutation



Csmith是C编译器最成功的随机测试系统。

在过去的几年中，它帮助找到了几百个编译器插件，并为改进GCC和LLVM的质量做出了重大贡献。

Program Mutation

Semantics-Preserving Mutation

主要思想是不改变程序行为的情况下对程序进行变异。

Non-Semantics-Preserving Mutation

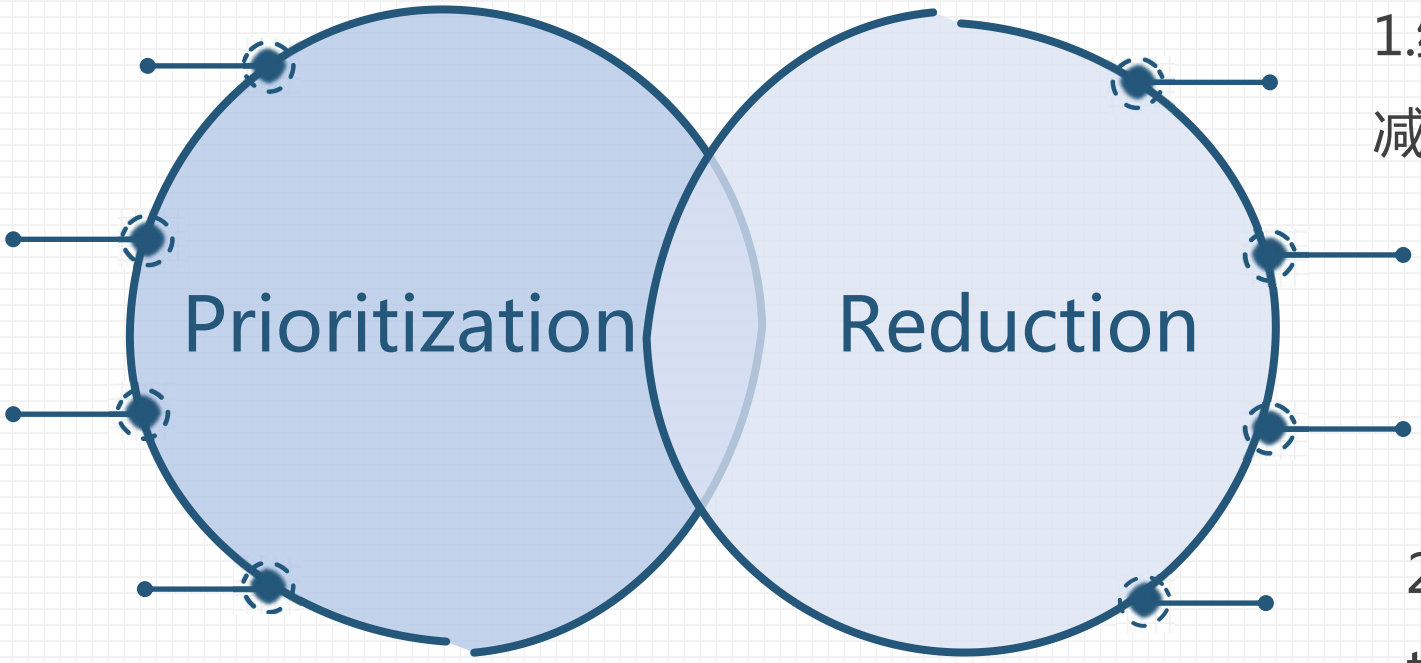
执行MCMC（马尔可夫链蒙特卡洛）采样，选择具有更大可能性触发编译器错误的突变。

Challenges

Validity of test programs.	如果为编译器提供了无效的输入程序，则该程序会在处理的初始阶段被丢弃。
Diversity of test programs	与所有测试的输入一样，构建的测试程序应多样化。这可能有助于发现错误。
Specific requirements imposed by a testing method	某些测试方法有特定的限制，此时测试程序的构建就会变得更困难。
Oracle	作为任何测试活动，编译器测试都必须解决test oracle问题，即确定测试程序是否触发了编译器的bug。

Optimizing The Test Process

确定测试程序的优先级



1.缩小测试输入程序从而减少与之相关的可疑文件

2.Increase passing test program reduce suspiciousness of innocent files

汇报完毕，恳请指正