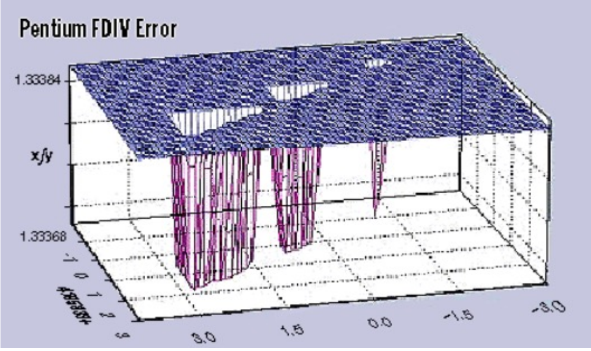
浙江大学
ZHEJIANG UNIVERSITY

蜕变测试导引

rainoftime
rainoftime.github.io
pyaoaa@zju.edu.cn

保障软件质量至关重要



Intel Pentium漏洞导致
声誉和巨额经济损失

Ariane5火箭升空数秒后爆炸
(损失85亿美元)

软件漏洞导致丰田回收120万
辆Prius汽车

纳斯达克OMX系统发生故障
造成千万美元损失

软件数据竞争问题导致美国
东北部大面积停电

数据竞争导致Therac25放疗
仪使用超过量的放射物

程序分析: 软件质量保证的重要手段

静态分析

动态分析

SonarQube Coverity Cppcheck PVS-Studio …

报告提纲

从测试预言问题到蜕变测试

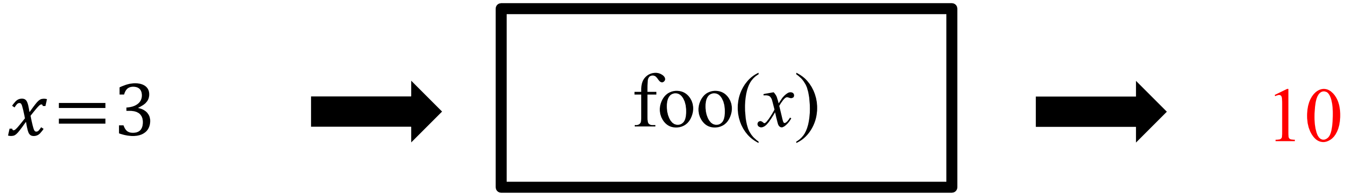
- **测试预言问题**
- 蜕变测试概述
- 蜕变关系的属性

面向SMT求解器的蜕变测试

- SMT求解器及其逻辑缺陷
- 基于语义融合的方法
- 基于近似枚举的方法

逻辑缺陷(正确性缺陷)

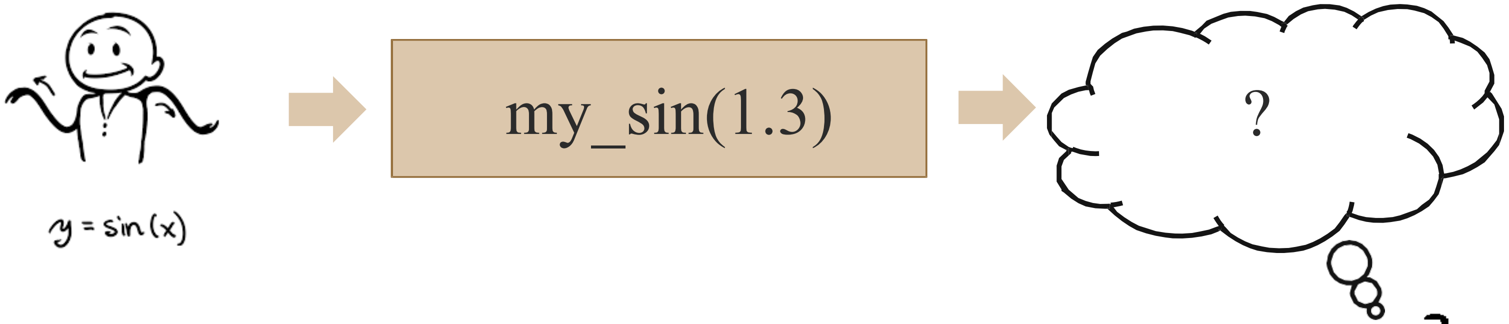
- 假设函数 foo(x)的目的是“计算并返回x的平方”



$x=3$ → foo(x) → 10

例: 测试my_sin的逻辑缺陷

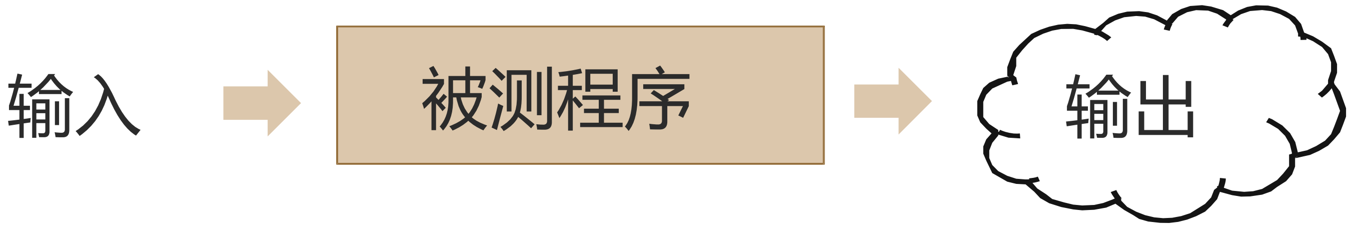
- 假设程序 my_sin 的目标是实现三角函数 *sin*



- 人工推算可能的结果?
- 对比其它工具的结果?

测试预言(Test Oracle)

给定测试输入，判断输出正确与否的机制



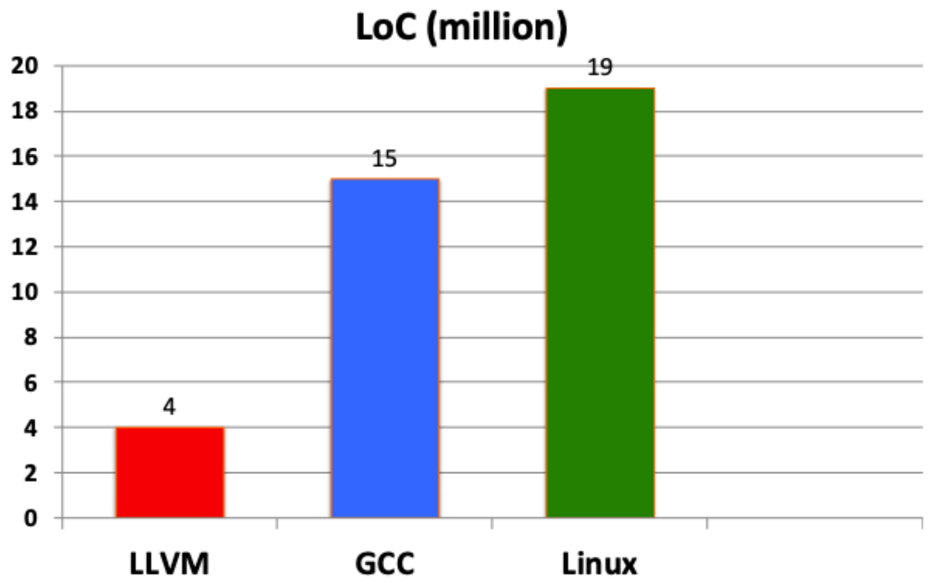
对发现逻辑缺陷至关重要

- my_sin 实现了 *sin* 函数吗?
- Coq证明了我的命题吗?

测试预言问题(Oracle Problem)

- Elaine Weyuker: “On Testing Non-testable Programs”, 1982

1. 不存在测试预言
2. 理论上存在, 但实际上很难 (自动) 检查



- GCC把Linux编译对了么?

报告提纲

从测试预言问题到蜕变测试

- 测试预言问题
- **蜕变测试概述**
- 蜕变关系的属性

面向SMT求解器的蜕变测试

- SMT求解器及其逻辑缺陷
- 基于语义融合的方法
- 基于近似枚举的方法

蜕变测试(Metamorphic Testing)

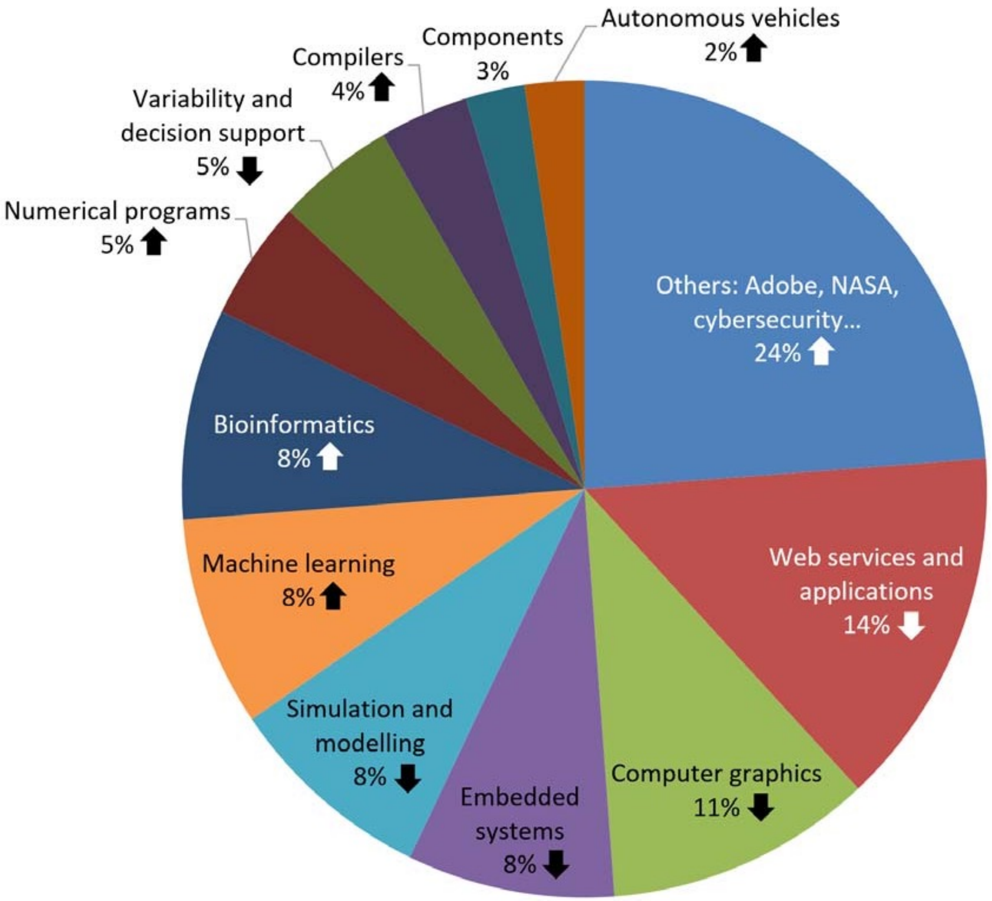
- 一类解决/缓解测试预言问题的方法 [T.Y. Chen et al., 1998]
- **主要想法:** 利用相关领域知识、交叉检查多组输入及其输出

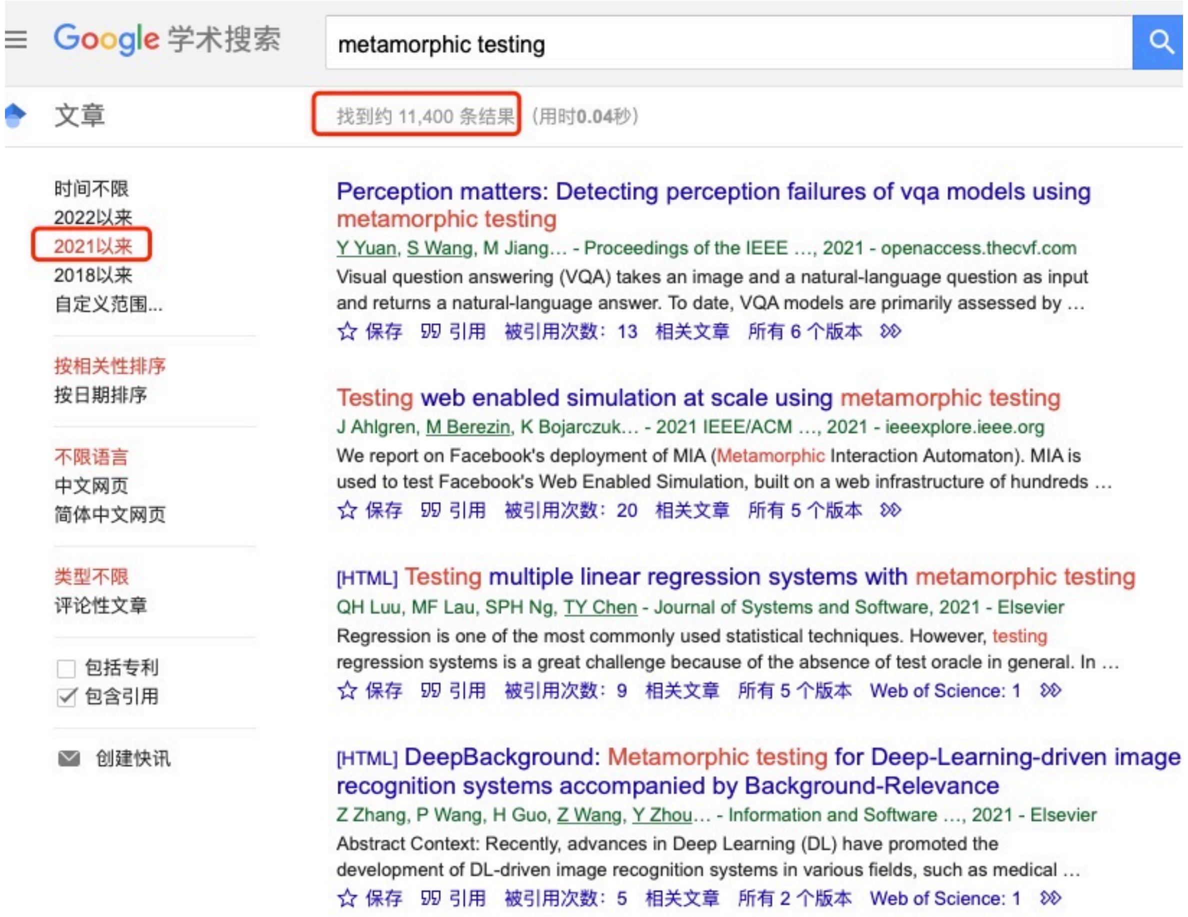
蜕变关系

Metamorphic Relation

虽然很难知道**各个具体输入的输出**应该是什么，
但可以利用**多组输入/输出之间的关系**

蜕变测试的应用





metamorphic testing

文章　找到约 11,400 条结果 (用时0.04秒)

时间不限
2022以来
2021以来
2018以来
自定义范围...

按相关性排序
按日期排序

不限语言
中文网页
简体中文网页

类型不限
评论性文章

□ 包括专利
☑ 包含引用

创建快讯

Perception matters: Detecting perception failures of vqa models using metamorphic testing
Y Yuan, S Wang, M Jiang… - Proceedings of the IEEE …, 2021 - openaccess.thecvf.com
Visual question answering (VQA) takes an image and a natural-language question as input and returns a natural-language answer. To date, VQA models are primarily assessed by …
☆ 保存 引用 被引用次数：13 相关文章 所有 6 个版本

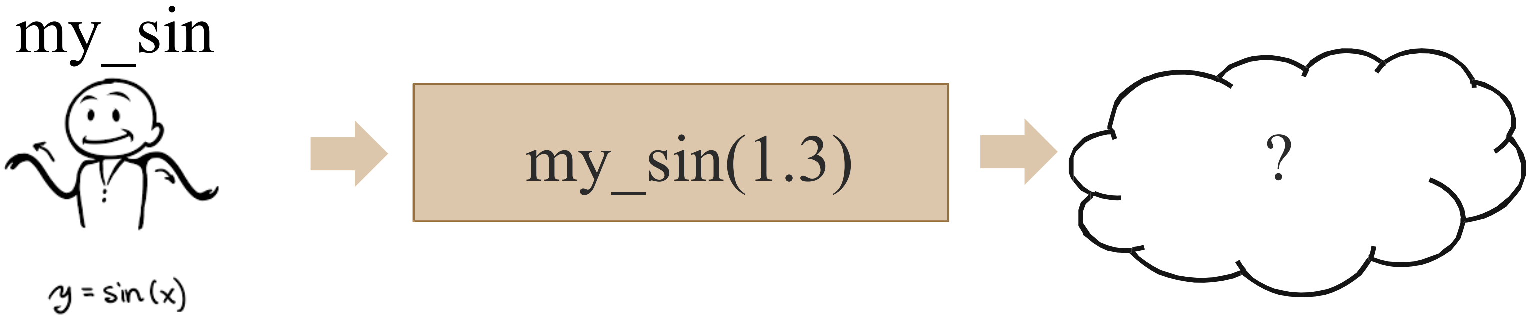
Testing web enabled simulation at scale using metamorphic testing
J Ahlgren, M Berezin, K Bojarczuk… - 2021 IEEE/ACM …, 2021 - ieeexplore.ieee.org
We report on Facebook's deployment of MIA (Metamorphic Interaction Automaton). MIA is used to test Facebook's Web Enabled Simulation, built on a web infrastructure of hundreds …
☆ 保存 引用 被引用次数：20 相关文章 所有 5 个版本

[HTML] Testing multiple linear regression systems with metamorphic testing
QH Luu, MF Lau, SPH Ng, TY Chen - Journal of Systems and Software, 2021 - Elsevier
Regression is one of the most commonly used statistical techniques. However, testing regression systems is a great challenge because of the absence of test oracle in general. In …
☆ 保存 引用 被引用次数：9 相关文章 所有 5 个版本 Web of Science: 1

[HTML] DeepBackground: Metamorphic testing for Deep-Learning-driven image recognition systems accompanied by Background-Relevance
Z Zhang, P Wang, H Guo, Z Wang, Y Zhou… - Information and Software …, 2021 - Elsevier
Abstract Context: Recently, advances in Deep Learning (DL) have promoted the development of DL-driven image recognition systems in various fields, such as medical …
☆ 保存 引用 被引用次数：5 相关文章 所有 2 个版本 Web of Science: 1

例: 测试my_sin函数

- 假设程序 my_sin 的目标是实现三角函数 *sin*

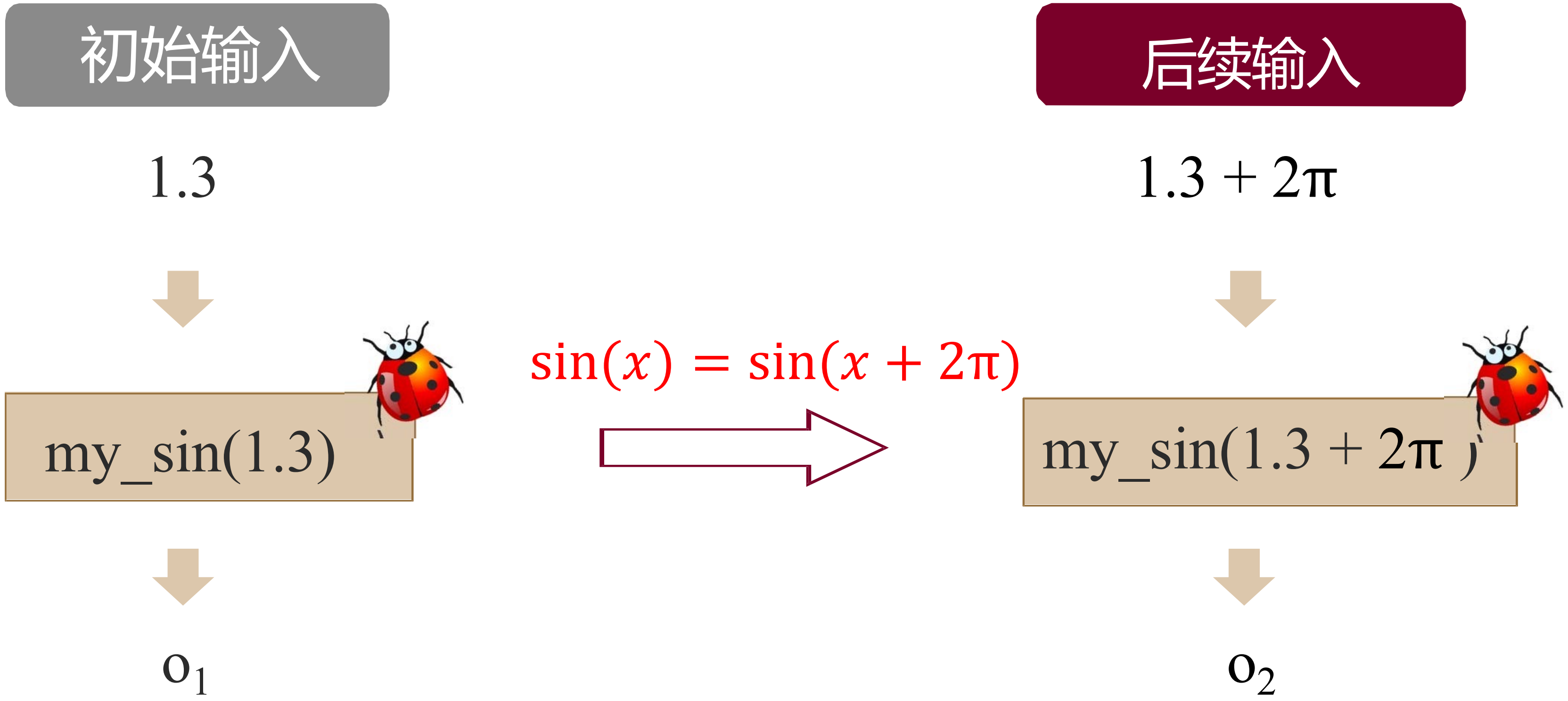

my_sin

my_sin(1.3)

?

- 不知道: my_sin(1.3) 应该返回什么?
- **但是知道**: $sin(x) = sin(x + 2\pi)$

例: 测试my_sin函数



初始输入

1.3

my_sin(1.3)

o_1

$\sin(x) = \sin(x + 2\pi)$

后续输入

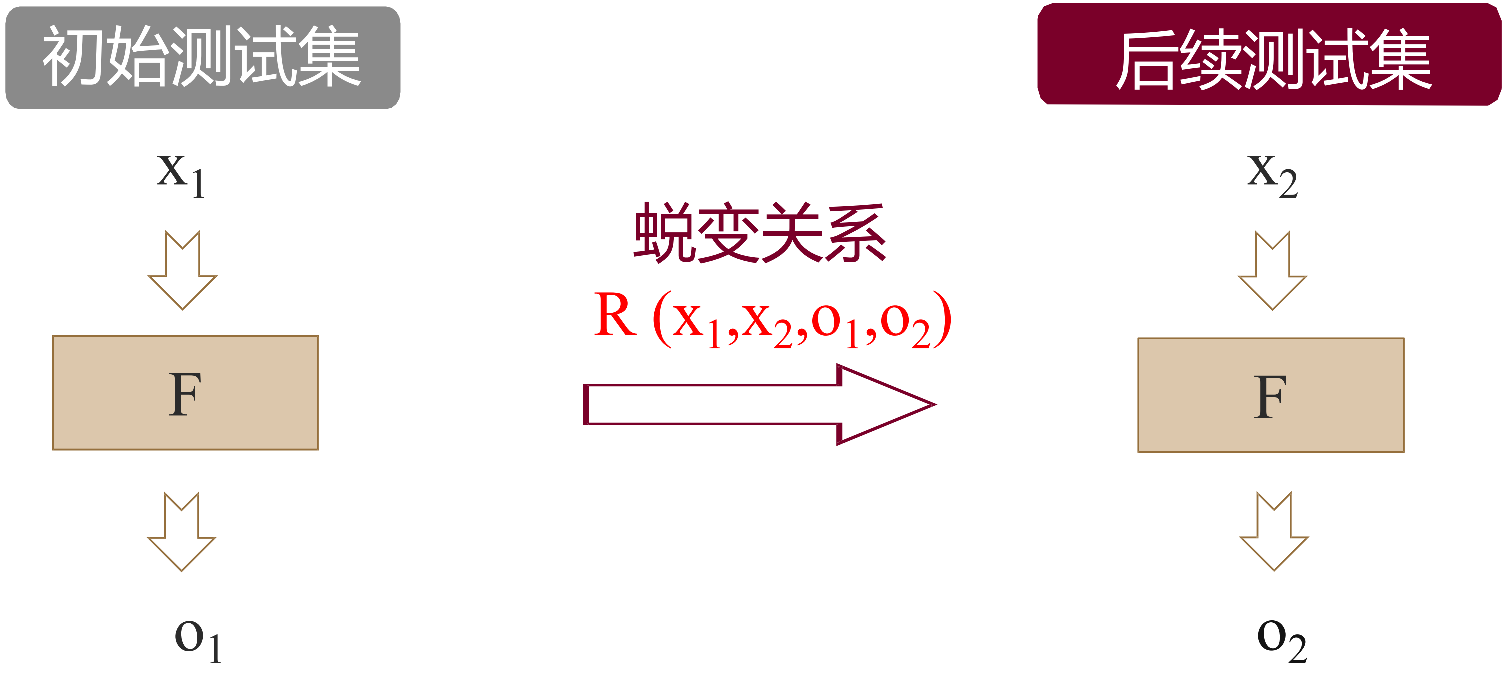
$1.3 + 2\pi$

my_sin($1.3 + 2\pi$)

o_2

检查: $\mathbf{o}_1 = \mathbf{o}_2$?

蜕变测试的基本要素



初始测试集

x_1

F

o_1

蜕变关系

R (x_1,x_2,o_1,o_2)

后续测试集

x_2

F

o_2

检查R是否被违背

蜕变测试工作流程

1. 发现/设计蜕变关系
2. 收集并运行初始测试集
3. 根据蜕变关系构造后续测试集
4. 检查是否违背蜕变关系

练习: 为my_sin填空

1 蜕变关系

sin(x) = sin(x + 2π)

2 起始测试集

1.2

3 构造后续测试集

1.2 + 2π

4 检查蜕变关系

my_sin(1.2) = my_sin(1.2 + 2π)?

报告提纲

从测试预言问题到蜕变测试

- 测试预言问题
- 蜕变测试概述
- **蜕变关系的属性**

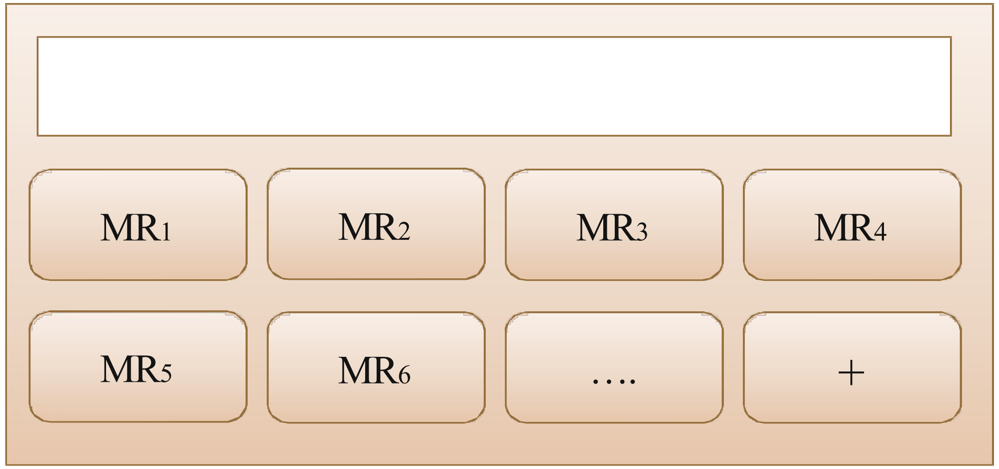
面向SMT求解器的蜕变测试

- SMT求解器及其逻辑缺陷
- 基于语义融合的方法
- 基于近似枚举的方法

讨论：蜕变关系是唯一的么?

一个被测程序可能有多个(甚至无穷个)蜕变关系

- $\sin(x) = \sin(x + 2\pi)$
- $\sin(x+\pi) = -\sin(x)$
- $\sin(\pi-x) = \sin(x)$

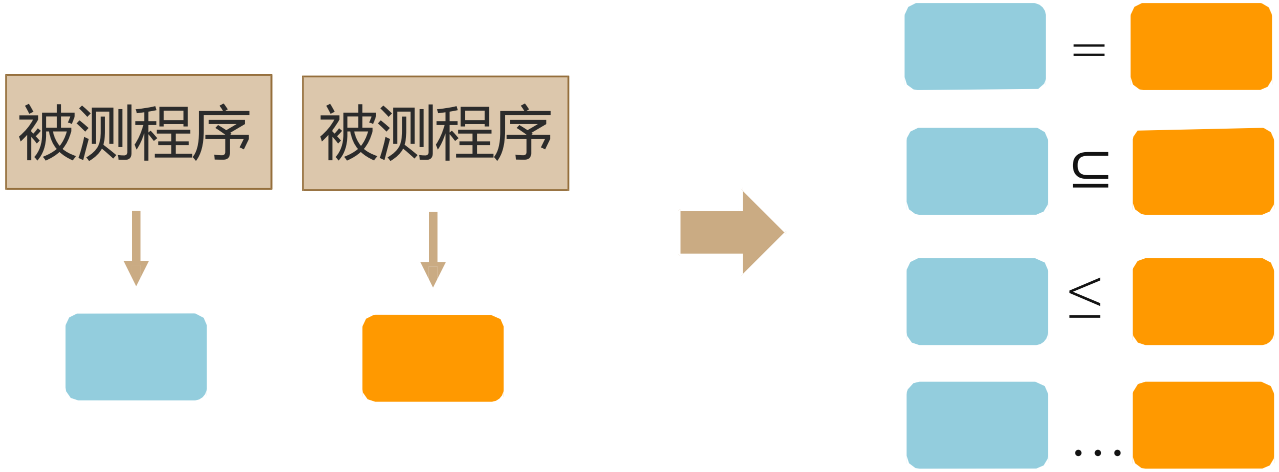


MR1 MR2 MR3 MR4

MR5 MR6 …. +

讨论: 蜕变关系 = 相等关系?

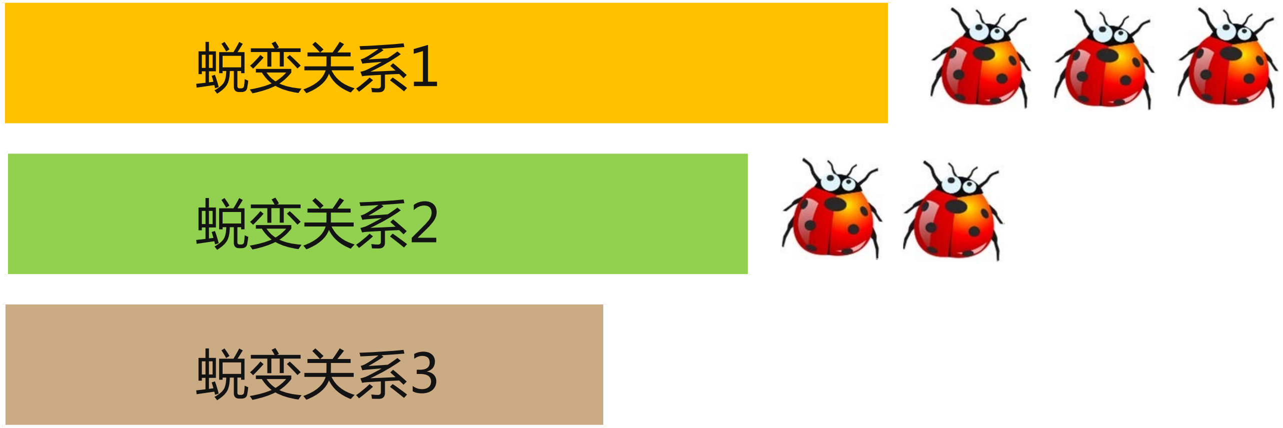
蜕变关系不限于相等关系, 也不限于数值关系



- 若$|x| \leq \pi/2$, 则 $x < y \Rightarrow \sin(x) < \sin(y)$

讨论: 每个蜕变关系都能找到Bug?

不同蜕变关系可能有不同的缺陷查找效果



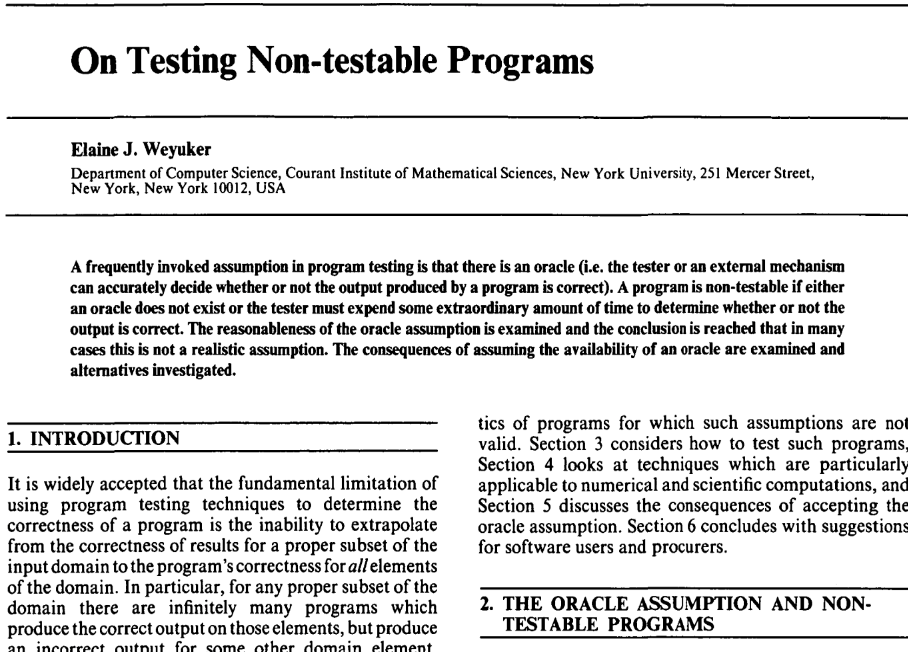
蜕变关系1

蜕变关系2

蜕变关系3

小结: 从测试预言问题到蜕变测试

测试预言和测试预言问题



On Testing Non-testable Programs

Elaine J. Weyuker

Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street,
New York, New York 10012, USA

A frequently invoked assumption in program testing is that there is an oracle (i.e. the tester or an external mechanism can accurately decide whether or not the output produced by a program is correct). A program is non-testable if either an oracle does not exist or the tester must expend some extraordinary amount of time to determine whether or not the output is correct. The reasonableness of the oracle assumption is examined and the conclusion is reached that in many cases this is not a realistic assumption. The consequences of assuming the availability of an oracle are examined and alternatives investigated.

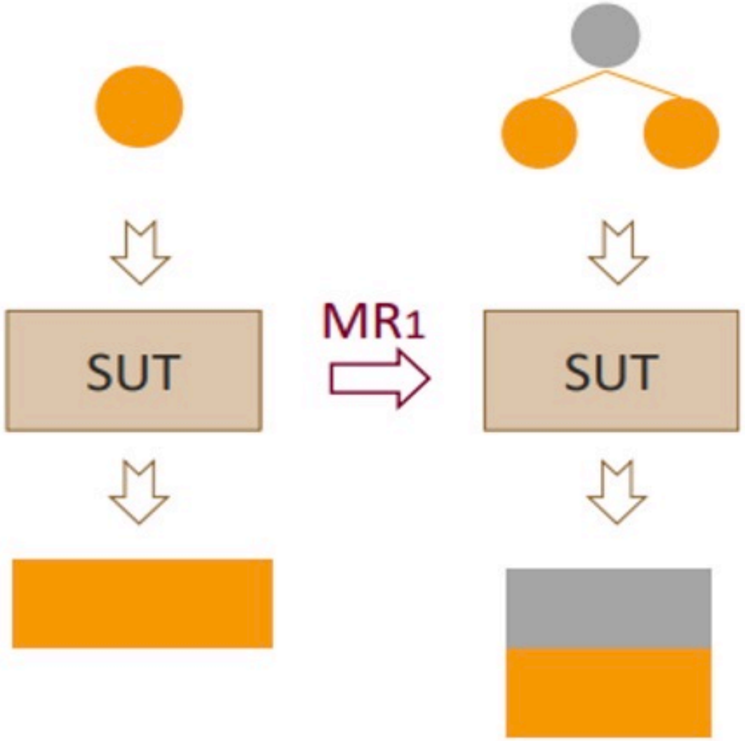
1. INTRODUCTION

It is widely accepted that the fundamental limitation of using program testing techniques to determine the correctness of a program is the inability to extrapolate from the correctness of results for a proper subset of the input domain to the program's correctness for *all* elements of the domain. In particular, for any proper subset of the domain there are infinitely many programs which produce the correct output on those elements, but produce an incorrect output for some other domain element

tics of programs for which such assumptions are not valid. Section 3 considers how to test such programs, Section 4 looks at techniques which are particularly applicable to numerical and scientific computations, and Section 5 discusses the consequences of accepting the oracle assumption. Section 6 concludes with suggestions for software users and procurers.

2. THE ORACLE ASSUMPTION AND NON-TESTABLE PROGRAMS

蜕变测试与蜕变关系



报告提纲

从测试预言问题到蜕变测试

- 测试预言和预言问题
- 蜕变测试概述
- 蜕变关系的属性

面向SMT求解器的蜕变测试

- **SMT求解器及其逻辑缺陷**
- 基于语义融合的方法
- 基于近似枚举的方法

Satisfiability Module Theories (SMT) 问题

$$\varphi : x > 0 \land x < 0$$

UNSAT

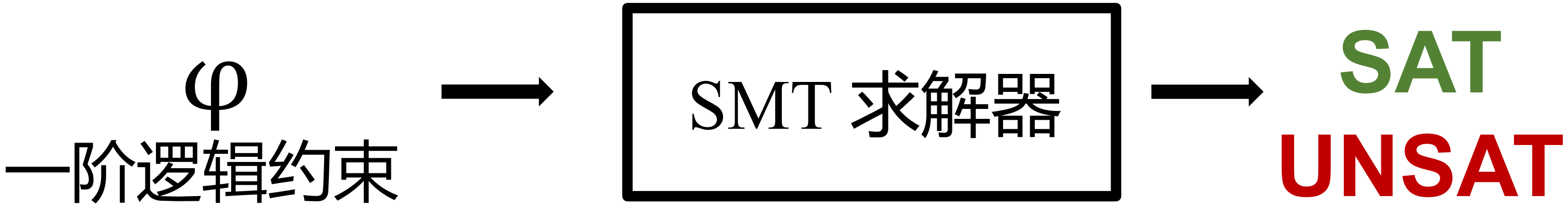
Satisfiability Module Theories (SMT) 问题

$$\varphi : x > 0 \land x < 1$$

SAT

Satisfiability Module Theories (SMT) 求解器

- 如果x是整数：$x * x = 3$ UNSAT
- 如果x是实数：$x * x = 3$ SAT



φ

一阶逻辑约束

SMT 求解器

SAT

UNSAT

Satisfiability Module Theories (SMT) 求解器

- **Z3求解器**：获得多个领域的重要奖项
  - 形式化方法
    - TACAS most influential paper award
    - ETAPS test-of-the-time award
    - CAV award
  - 编程语言
    - SIGPLAN software award
  - 自动推理
    - Skolem award
    - Herbrand award

SMT求解器的一些应用

- 寻找可满足解
  - 生成测试用例 [OSDI’08 最佳论文]
  - 生成线程调度 [PLDI’13 杰出论文(亚洲首次)]

KLEE

CLAP

- 证明不可满足
  - 检查验证条件 [POPL’02, 软件模型检验“开山之作”]
  - 检查类型签名 [ICFP’14, “自带SMT求解器的编译器”]

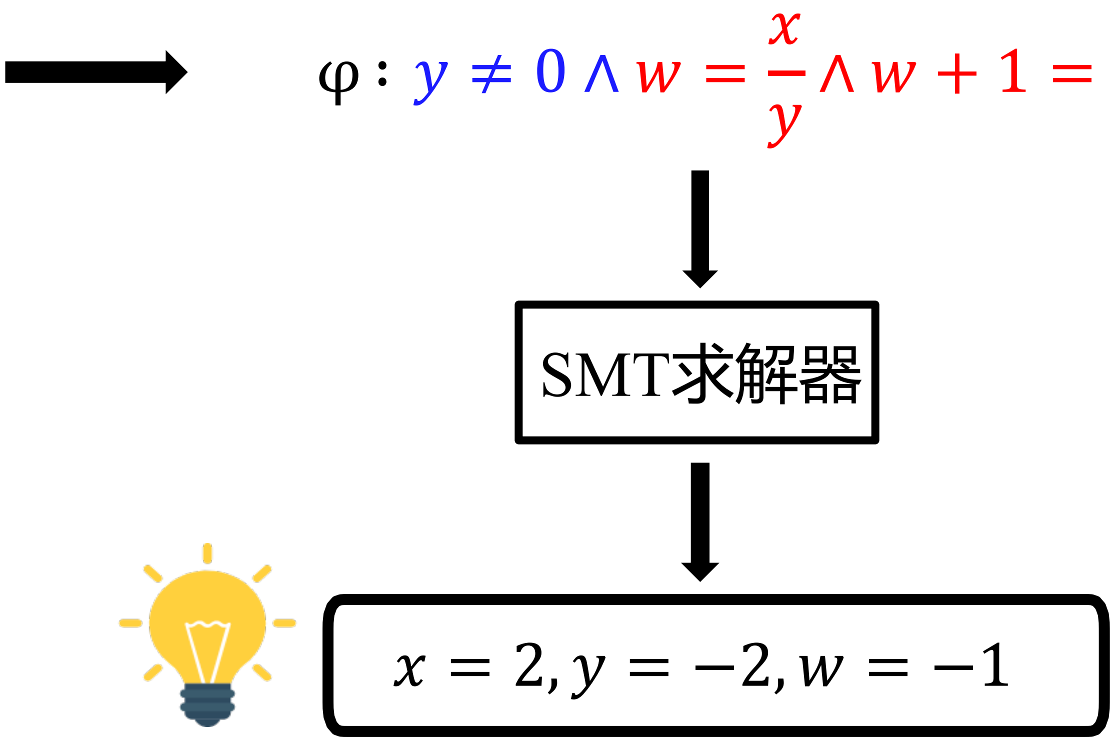
SLAM

LiquidHaskell

亚马逊每天调用数千万次Z3和CVC4 SMT求解器![1]

例: 基于SMT求解的程序分析

```
1 int main() {
2   int x, y = input();
3   if(y != 0){
4       int w = x / y;
5       print(10 / (w + 1));
6   }
7 }
```



$$\varphi: y \neq 0 \wedge w = \frac{x}{y} \wedge w + 1 = 0$$

SMT求解器

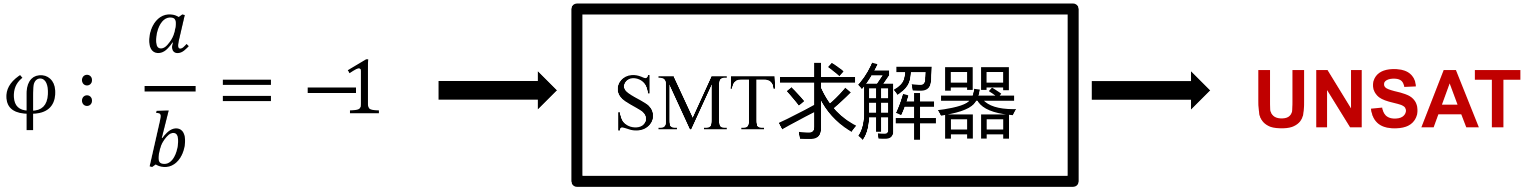
$$x = 2, y = -2, w = -1$$

第5行可能有除零错误吗?

港科大近期相关成果:

Guo et al., Precise Divide-By-Zero Detection with Affirmative Evidence, ICSE’22

SMT求解器的逻辑缺陷

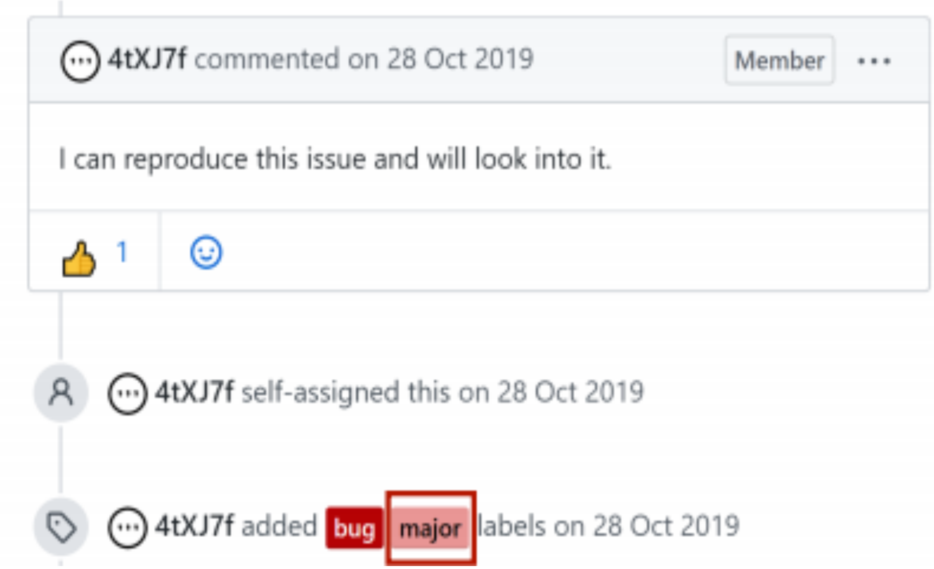


$\varphi : \dfrac{a}{b} = -1 \longrightarrow \boxed{\text{SMT求解器}} \longrightarrow$ **UNSAT**

```
% cat formula.smt2
(declare-fun a () Int)
(declare-fun b () Int)
(assert (= (div a b) (- 1)))
(check-sat)

% z3 formula.smt2
sat

% cvc4 formula.smt2
unsat
```



4tXJ7f commented on 28 Oct 2019 Member ···

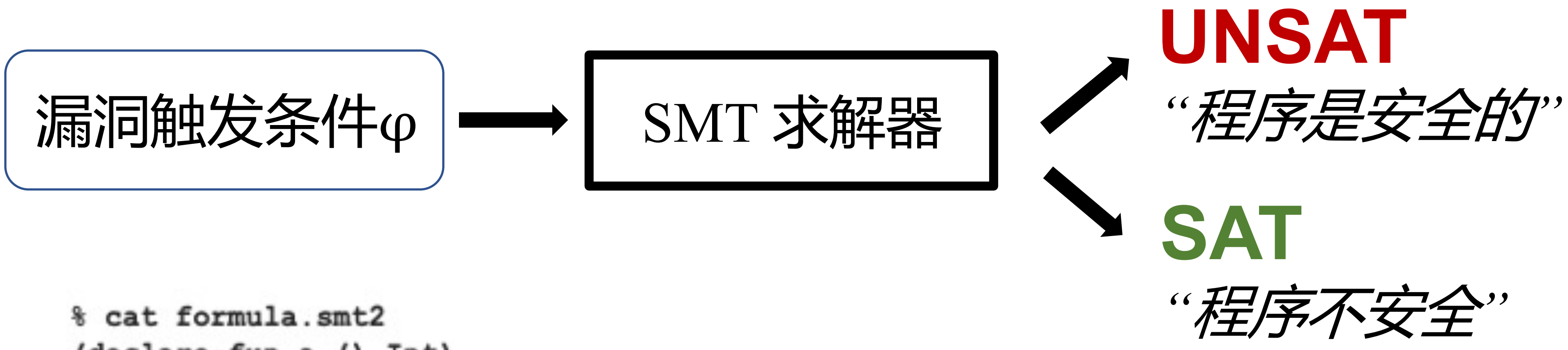
I can reproduce this issue and will look into it.

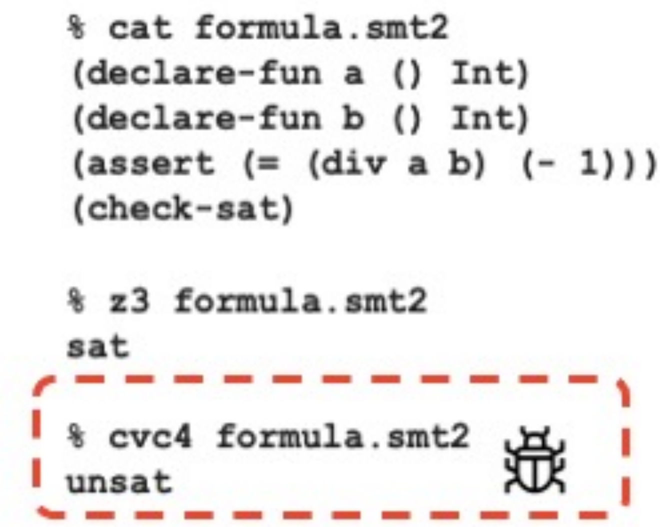
👍 1

4tXJ7f self-assigned this on 28 Oct 2019

4tXJ7f added bug major labels on 28 Oct 2019

SMT求解器逻辑缺陷的影响





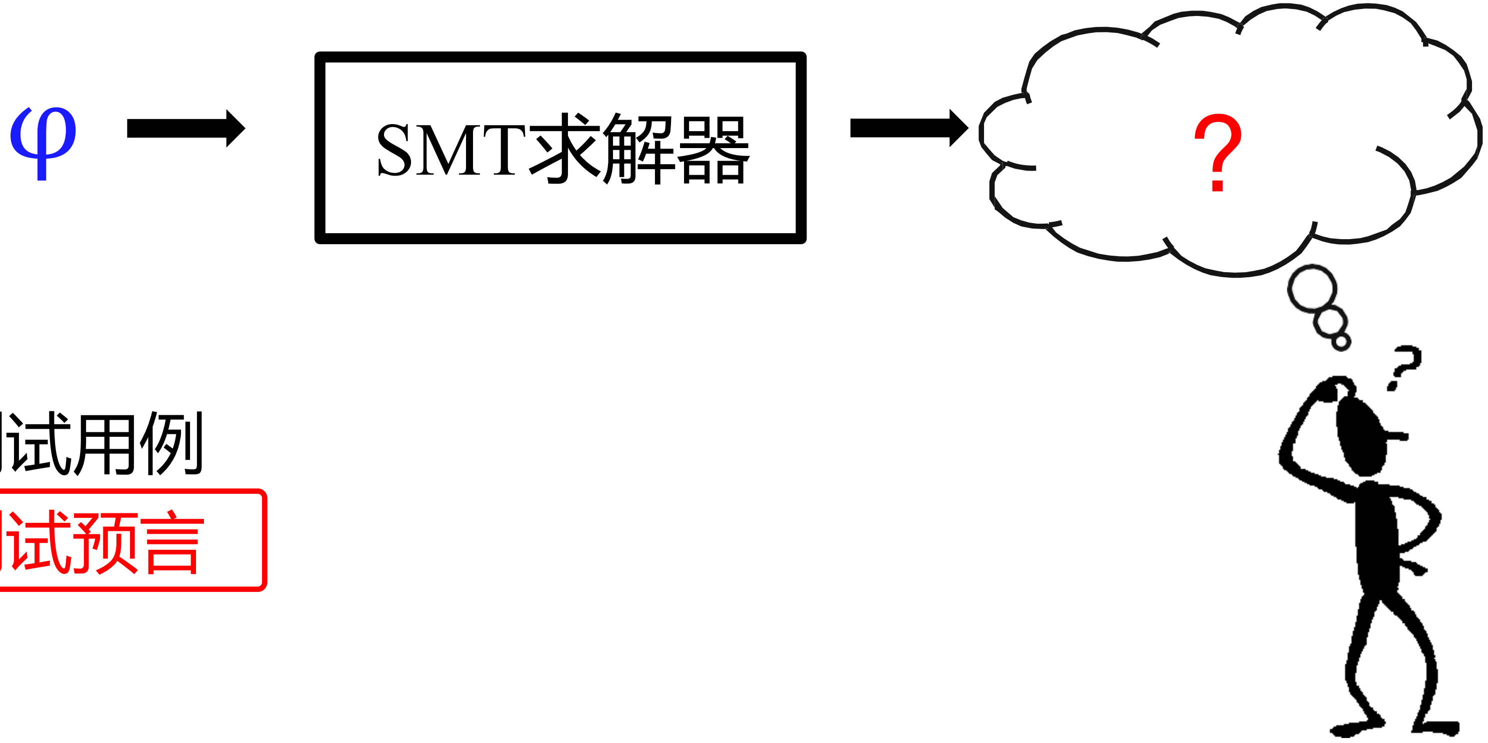
```
% cat formula.smt2
(declare-fun a () Int)
(declare-fun b () Int)
(assert (= (div a b) (- 1)))
(check-sat)

% z3 formula.smt2
sat

% cvc4 formula.smt2
unsat
```

- 分析结果变得不可信
- 可能错过关键漏洞

SMT求解器自动测试的挑战



φ

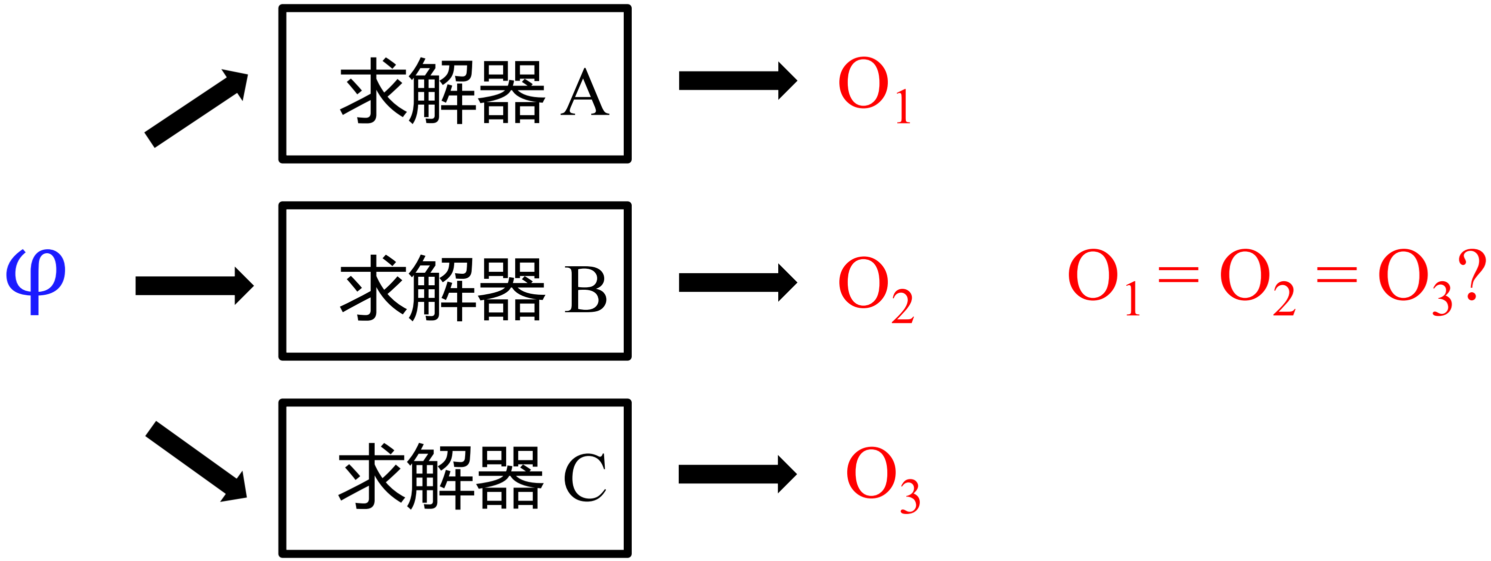
SMT求解器

?

1. 如何获得测试用例
2. 如何获得测试预言

获得测试预言的可能方案

差分测试(differential testing) e.g., [SMT’09, CAV’18, OOPSLA’20, …]



φ

求解器 A

求解器 B

求解器 C

O_1

O_2

O_3

$O_1 = O_2 = O_3$?

局限: 不能处理特殊类型的约束(比如只有求解器A支持)

报告提纲

蜕变测试方法简介

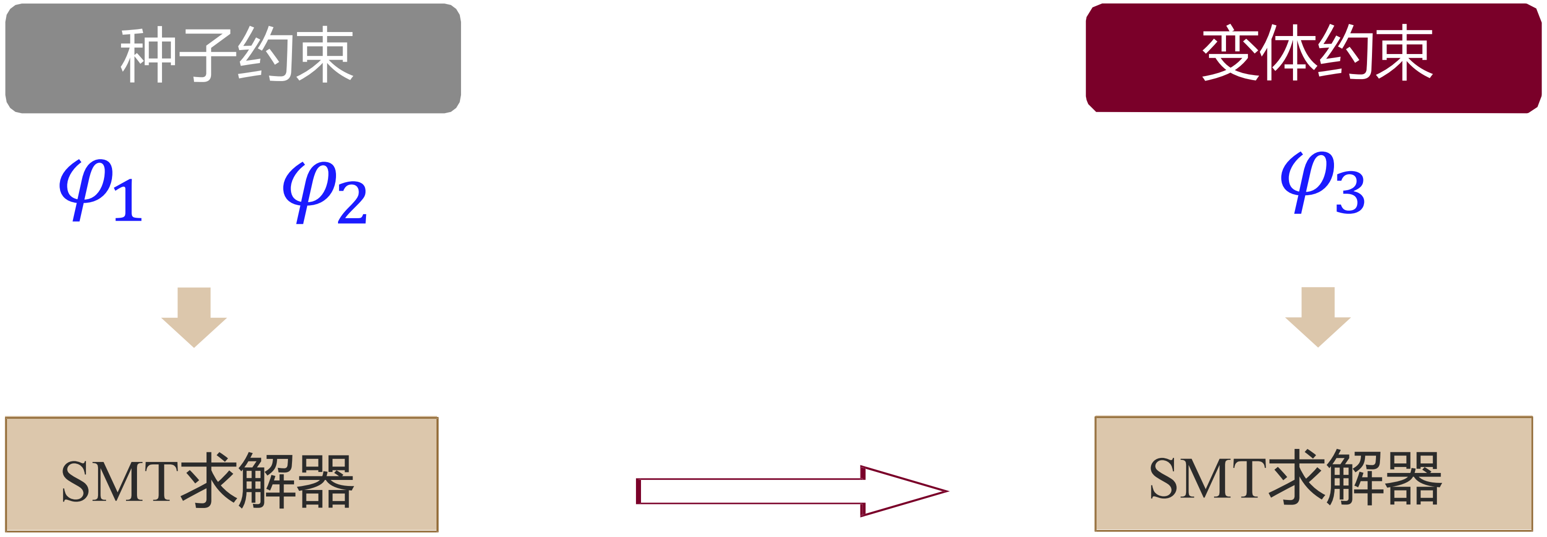
- 测试预言问题
- 蜕变测试概述
- 蜕变关系的属性

面向SMT求解器的蜕变测试

- SMT求解器及其逻辑缺陷
- **基于语义融合的方法**
- 基于近似枚举的方法

YinYang: 基于语义融合的蜕变测试 [Winterer et al. PLDI’20]

- “融合” 已知可满足性的种子约束, 得到带预期结果的新约束



种子约束

φ_1 φ_2

SMT求解器

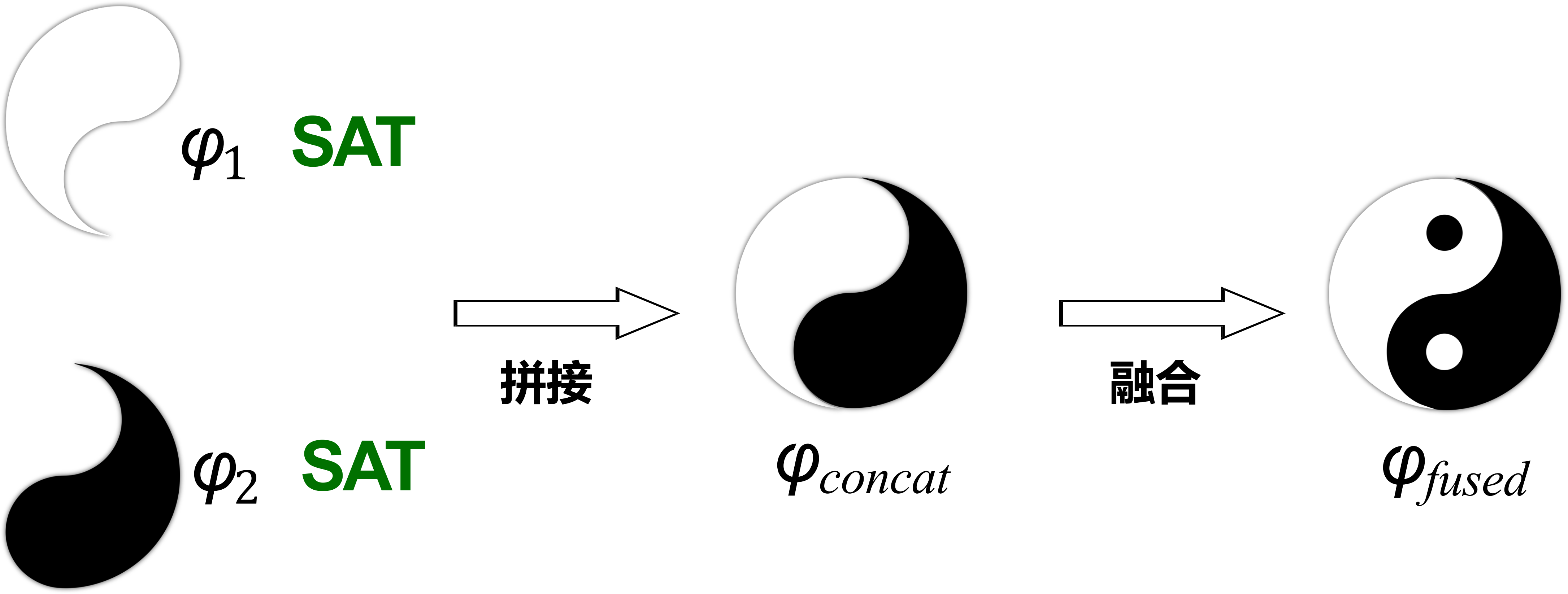
变体约束

φ_3

SMT求解器

Winterer et. al., Validating SMT Solvers via Semantic Fusion, PLDI’20 🏆

语义融合: 方法流程



φ_1 SAT

φ_2 SAT

拼接

φ_{concat}

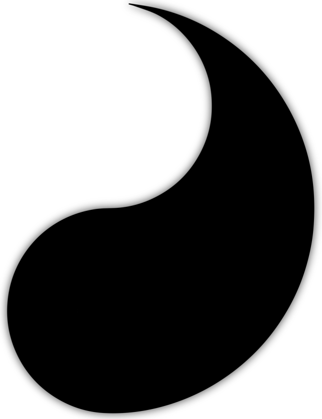
融合

φ_{fused}

例: 拼接两个约束

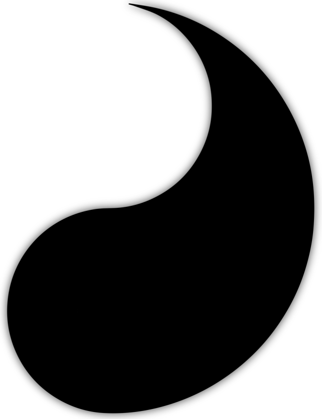
$\varphi_1 = x > 0 \wedge x > 1$ **SAT**

$\varphi_2 = y < 0 \wedge y < 1$ **SAT**



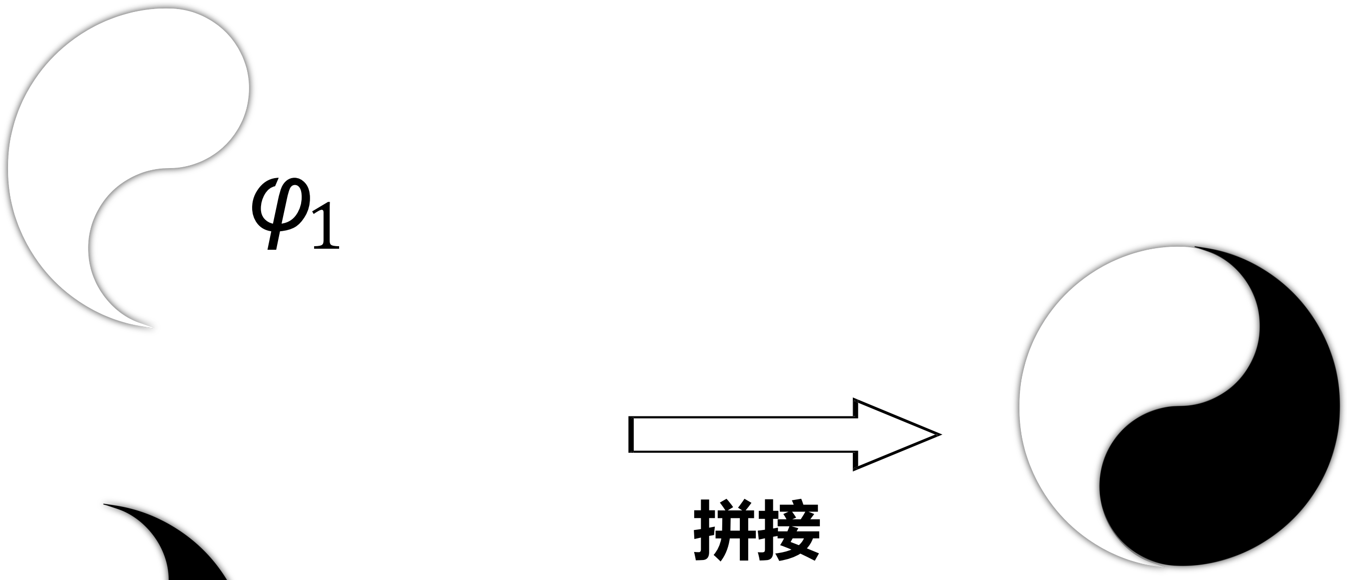
例: 拼接两个约束

$$\boxed{\varphi_{concat}} = \overset{\varphi_1}{(x > 0 \wedge x > 1)} \wedge \overset{\varphi_2}{(y < 0 \wedge y < 1)}$$ **SAT**



$$\boxed{x = 2 \qquad\qquad y = -2}$$

讨论: $\varphi_1, \varphi_2, \varphi_{concat}$是否构成某种蜕变关系?



φ_1

φ_2

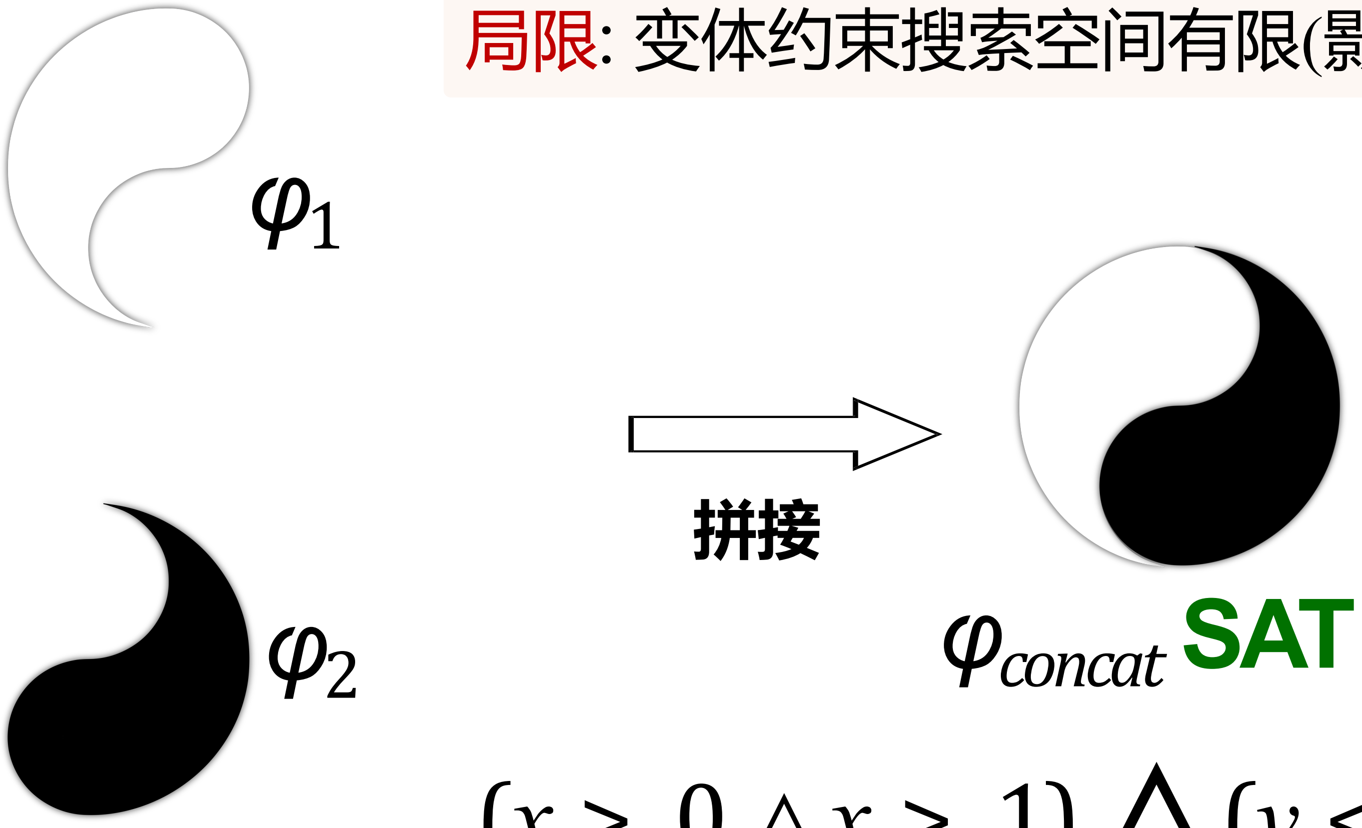
拼接

φ_{concat} **SAT**

$$\underbrace{(x > 0 \wedge x > 1)}_{\varphi_1} \wedge \underbrace{(y < 0 \wedge y < 1)}_{\varphi_2}$$

讨论: 直接把 φ_{concat} 作为变体约束?

局限: 变体约束搜索空间有限(影响方法效果)



φ_1

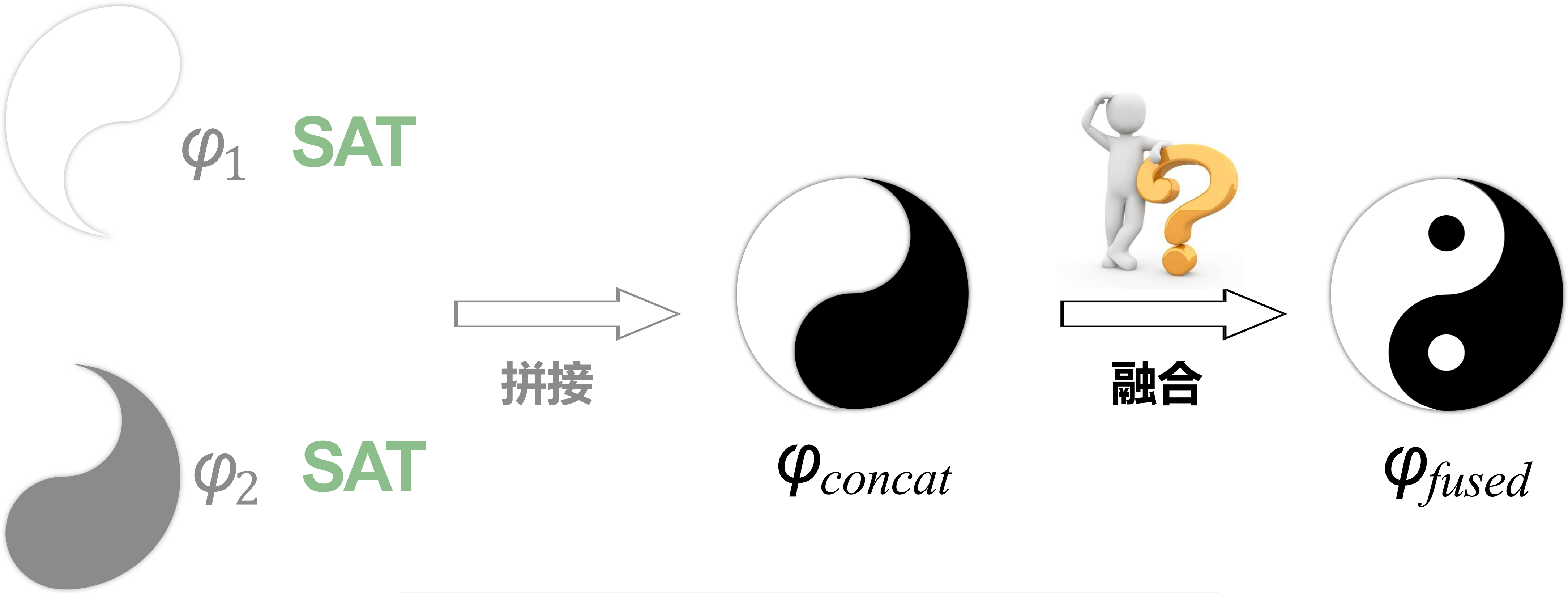
φ_2

拼接

φ_{concat} SAT

$$\underset{\varphi_1}{(x > 0 \wedge x > 1)} \bigwedge \underset{\varphi_2}{(y < 0 \wedge y < 1)}$$

关键挑战: 如何进一步做融合?



φ_1 SAT

φ_2 SAT

拼接

φ_{concat}

融合

φ_{fused}

目标：形式多样性; 可满足性明确

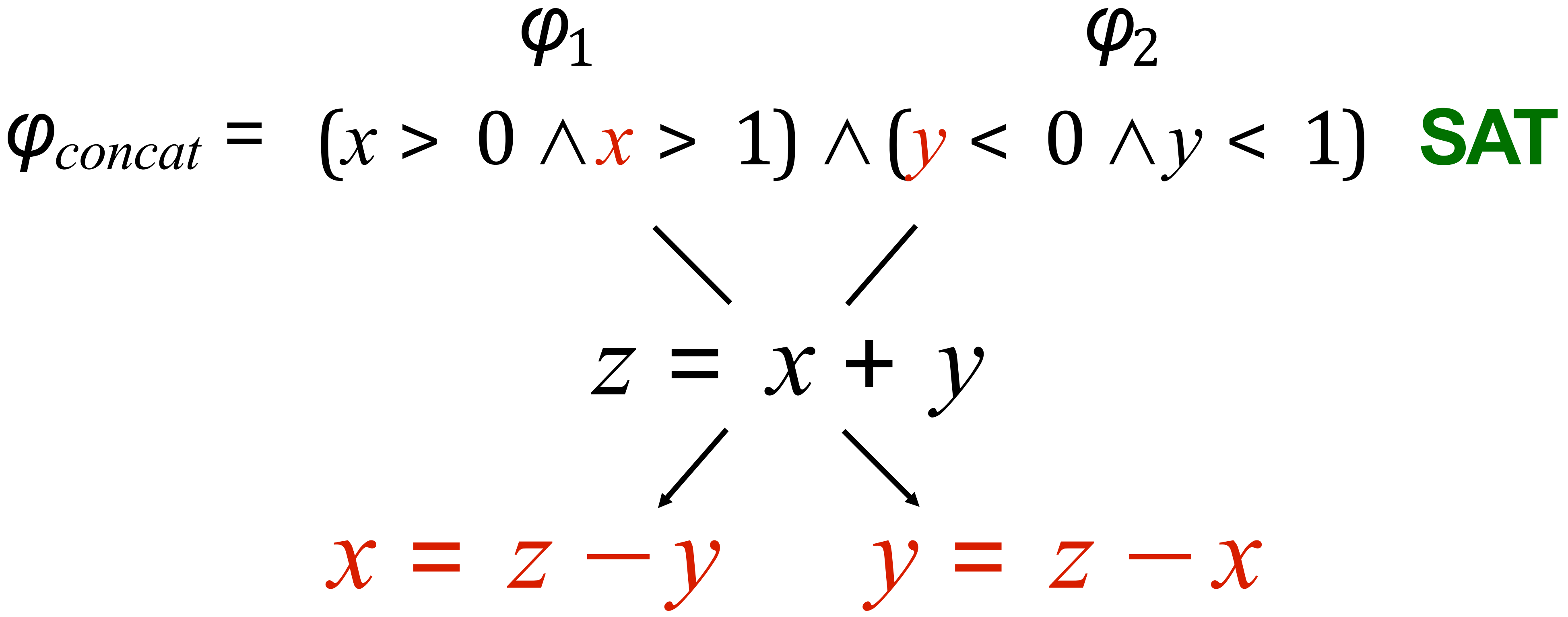
解决方案: 融合函数(Fusion Function)

$$\varphi_{concat} = \overset{\varphi_1}{(x > 0 \land x > 1)} \land \overset{\varphi_2}{(y < 0 \land y < 1)}$$ **SAT**

$$z = x + y$$

Fusion Function

构造融合函数的逆函数



$$\varphi_{concat} = \overset{\varphi_1}{(x > 0 \wedge x > 1)} \wedge \overset{\varphi_2}{(y < 0 \wedge y < 1)}$$

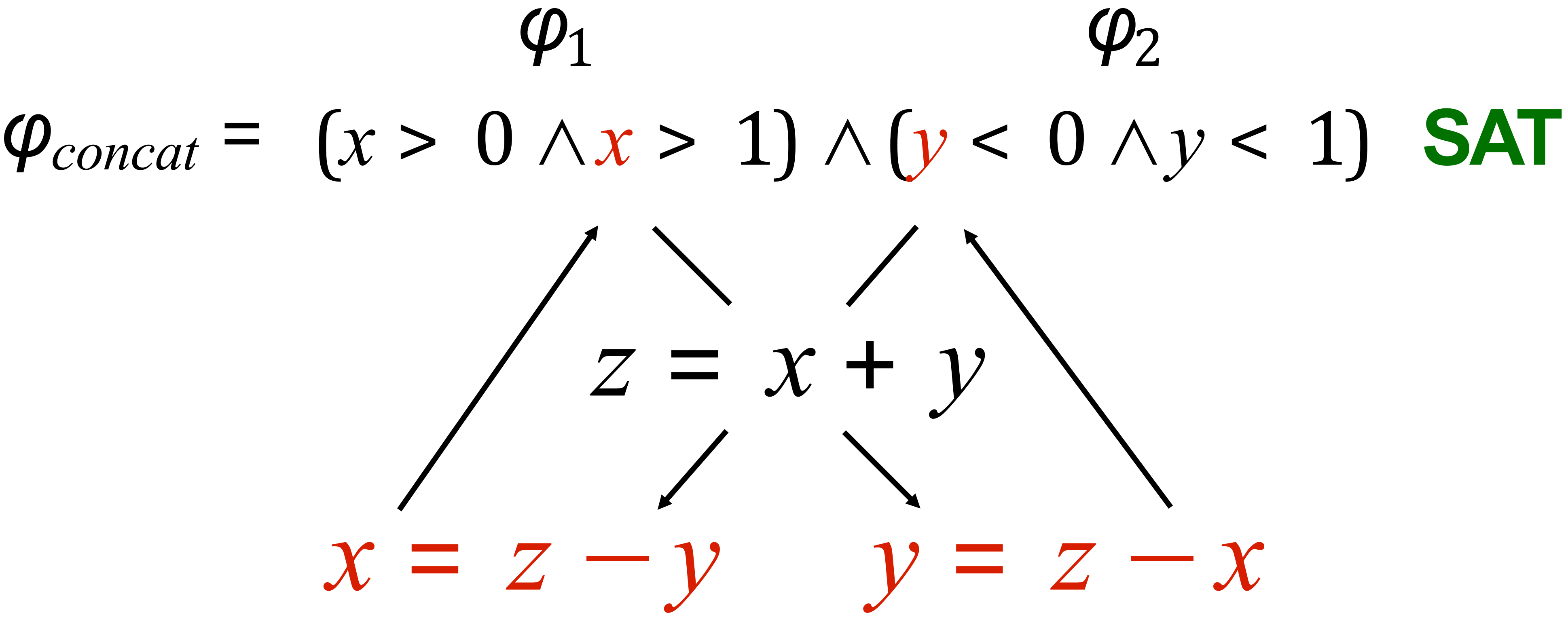
SAT

$$z = x + y$$

$$x = z - y \qquad y = z - x$$

Inversion Function

利用逆函数做变量替换



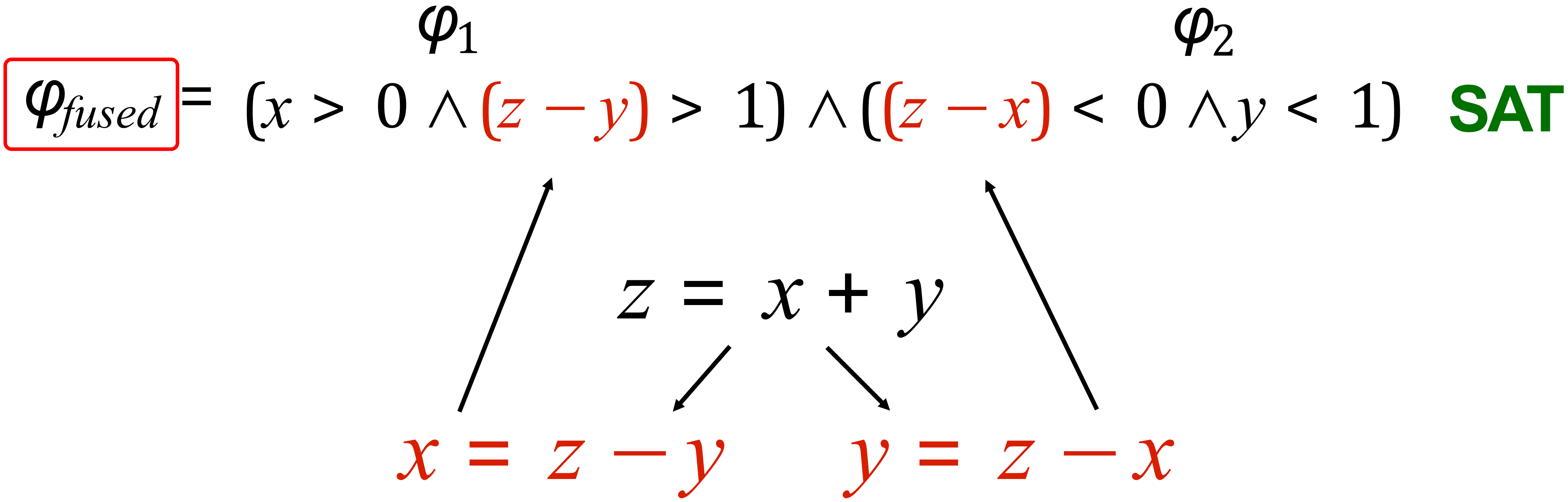
φ_1 φ_2

$\varphi_{concat} = (x > 0 \wedge x > 1) \wedge (y < 0 \wedge y < 1)$ **SAT**

$z = x + y$

$x = z - y$ $y = z - x$

利用逆函数做变量替换



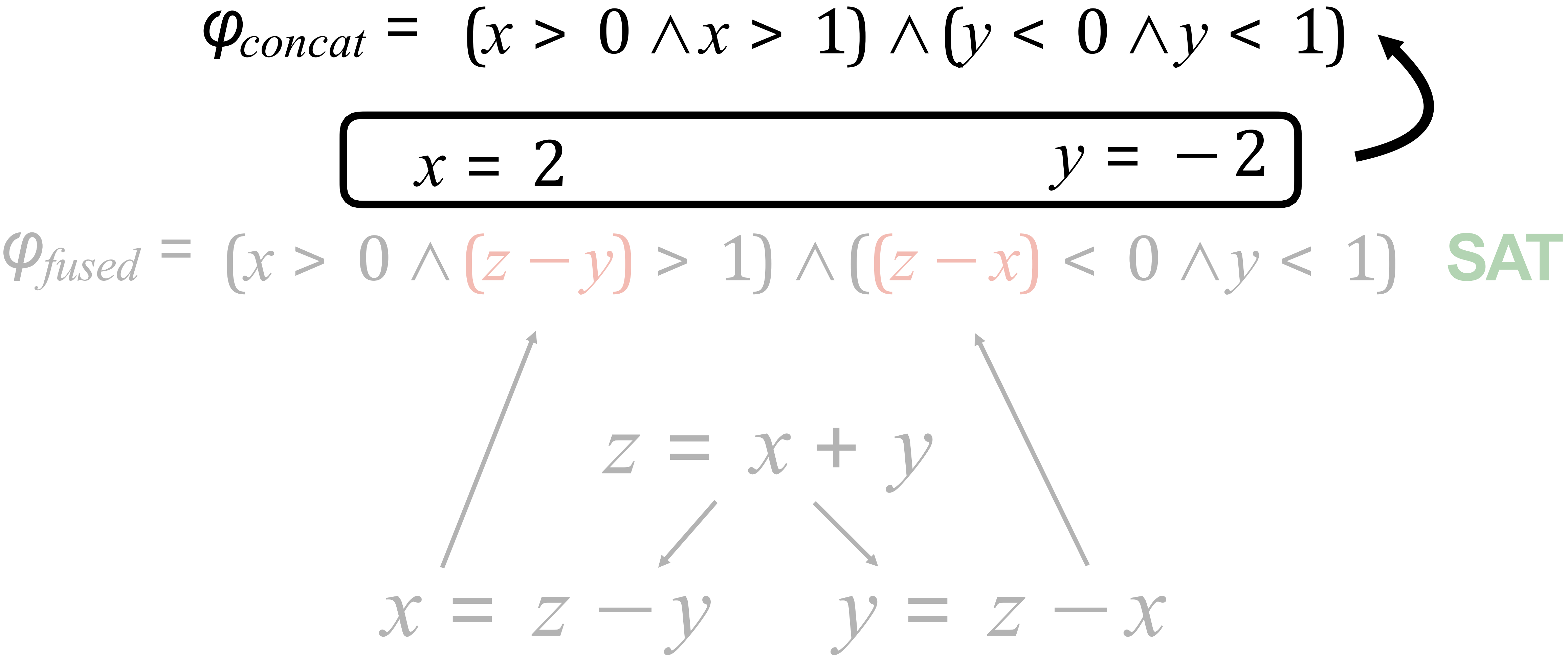
$$\boxed{\varphi_{fused}} = \overset{\varphi_1}{(x > 0 \wedge (z - y) > 1)} \wedge \overset{\varphi_2}{((z - x) < 0 \wedge y < 1)}$$

SAT

$$z = x + y$$

$$x = z - y \qquad y = z - x$$

φ_{fused}保留了φ_{concat}的可满足性



$$\varphi_{concat} = (x > 0 \land x > 1) \land (y < 0 \land y < 1)$$

$$\boxed{x = 2 \qquad y = -2}$$

$$\varphi_{fused} = (x > 0 \land (z - y) > 1) \land ((z - x) < 0 \land y < 1)$$

SAT

$$z = x + y$$

$$x = z - y \qquad y = z - x$$

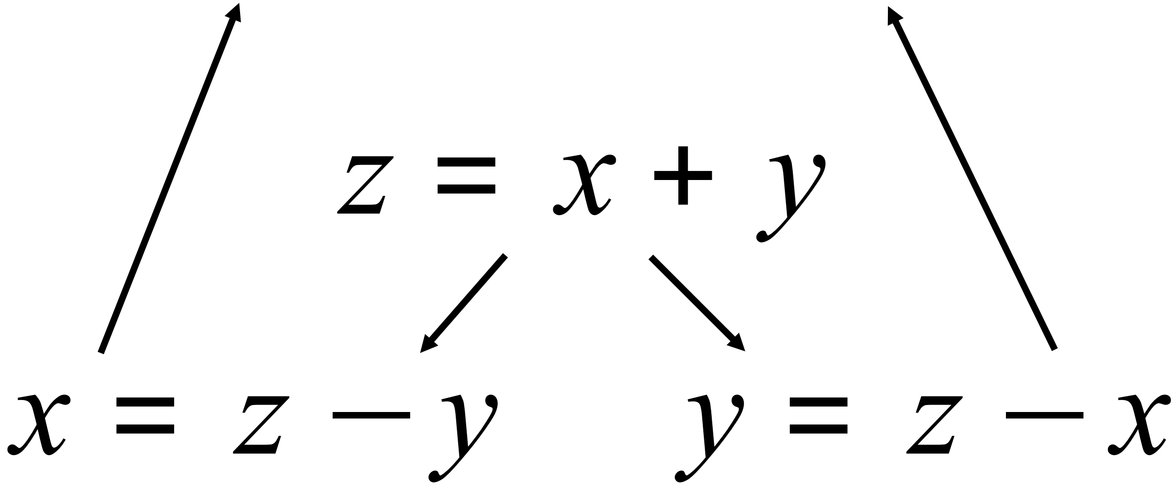
φ_{fused}保留了φ_{concat}的可满足性

$$\varphi_{concat} = (x > 0 \wedge x > 1) \wedge (y < 0 \wedge y < 1)$$

$$\boxed{x = 2 \quad z = x + y = 0 \quad y = -2}$$

$$\varphi_{fused} = (x > 0 \wedge (z - y) > 1) \wedge ((z - x) < 0 \wedge y < 1)$$

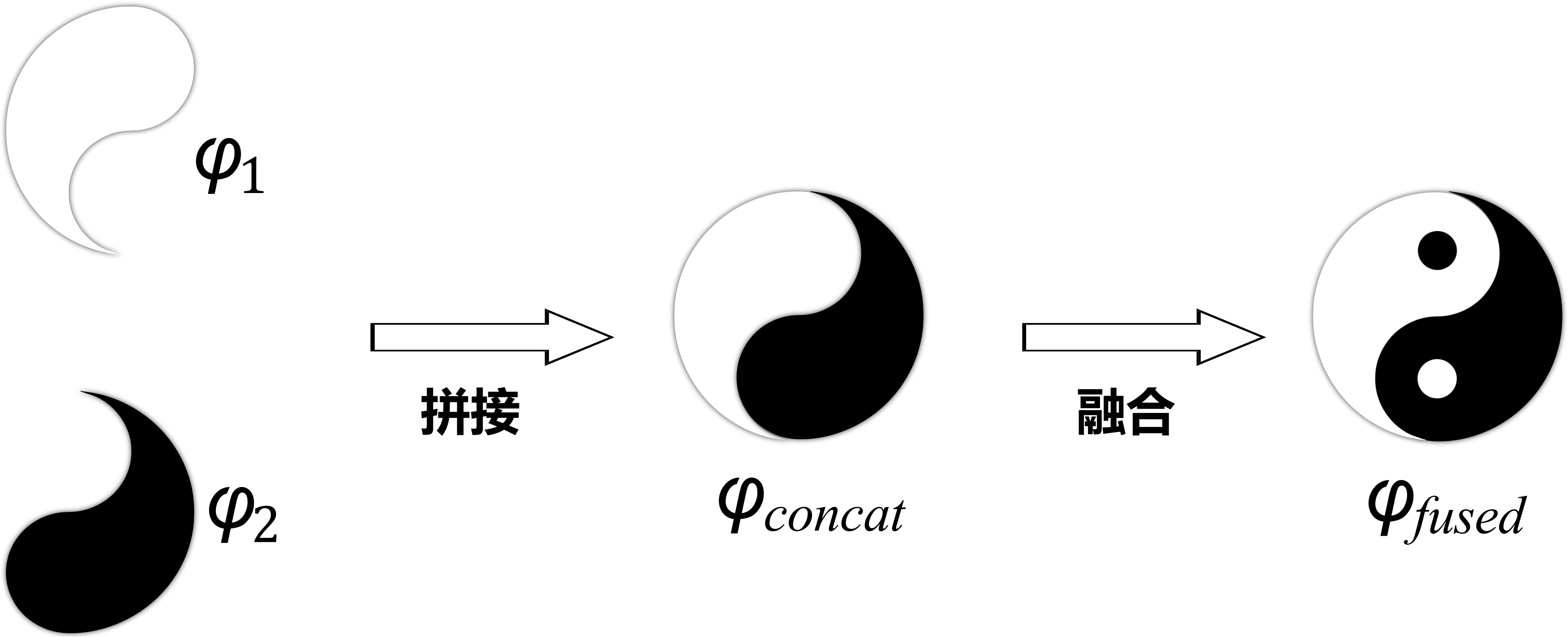
SAT



$$z = x + y$$

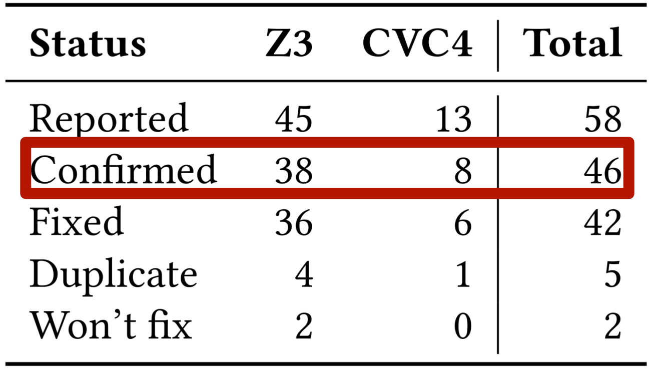
$$x = z - y \qquad y = z - x$$

基于语义融合的方法

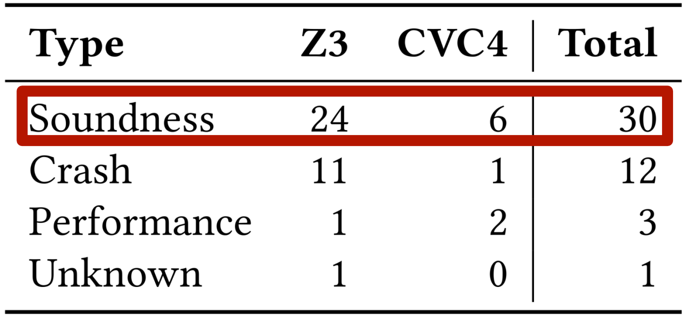


YinYang部分实验结果

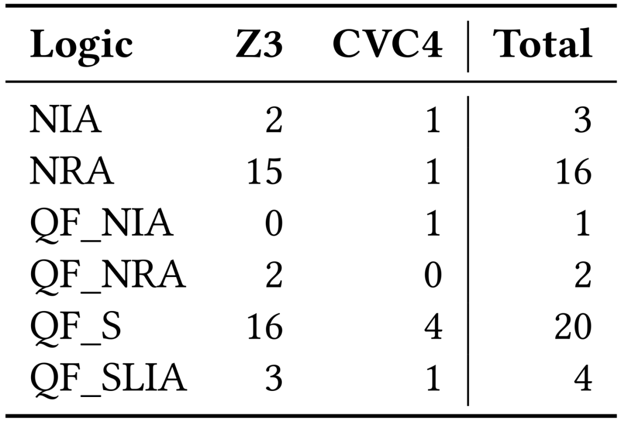
- 测试时间: 2019.07-2019.10



Status	Z3	CVC4	Total
Reported	45	13	58
Confirmed	38	8	46
Fixed	36	6	42
Duplicate	4	1	5
Won’t fix	2	0	2



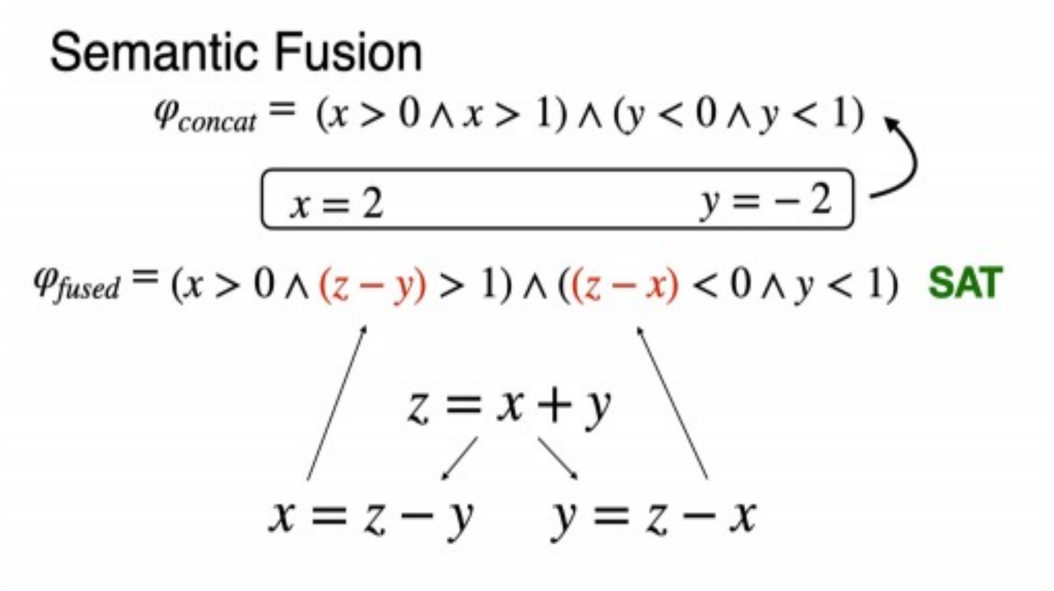
Type	Z3	CVC4	Total
Soundness	24	6	30
Crash	11	1	12
Performance	1	2	3
Unknown	1	0	1



Logic	Z3	CVC4	Total
NIA	2	1	3
NRA	15	1	16
QF_NIA	0	1	1
QF_NRA	2	0	2
QF_S	16	4	20
QF_SLIA	3	1	4

语义融合方法的一些特征

需要知道种子约束的可满足性



Semantic Fusion

$$\varphi_{concat} = (x > 0 \land x > 1) \land (y < 0 \land y < 1)$$

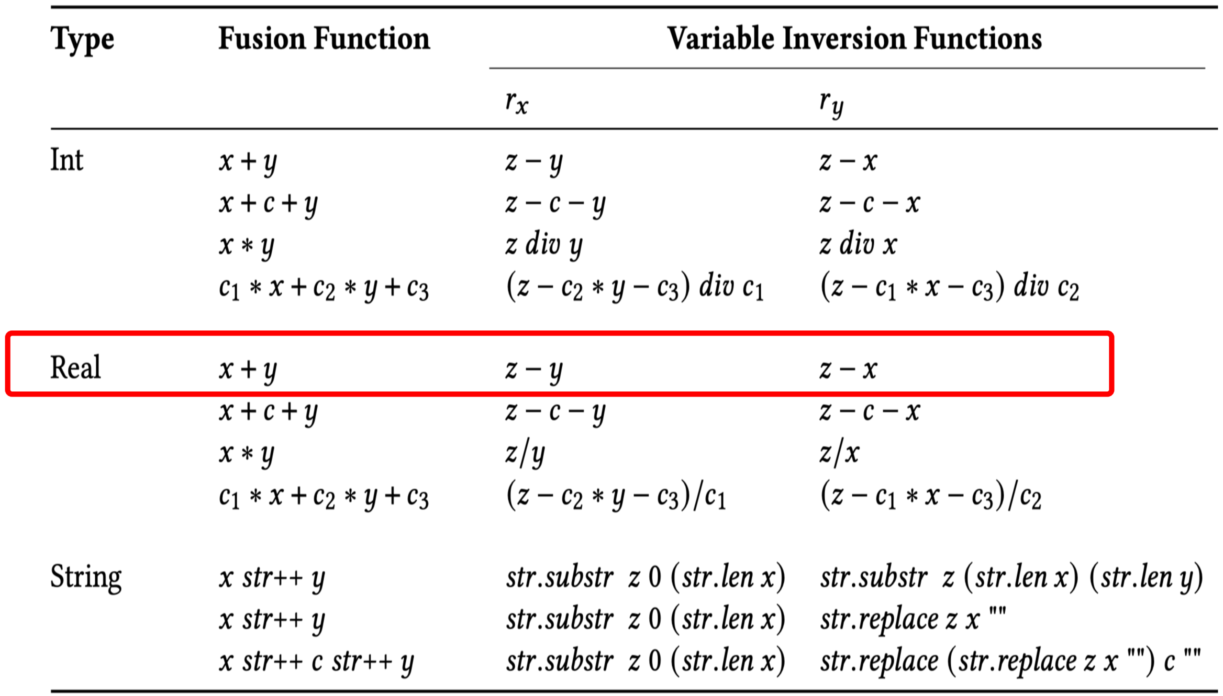
$$\boxed{x = 2 \qquad y = -2}$$

$$\varphi_{fused} = (x > 0 \land (z - y) > 1) \land ((z - x) < 0 \land y < 1) \quad \text{SAT}$$

$$z = x + y$$

$$x = z - y \qquad y = z - x$$

融合函数限制了**变体约束搜索空间**



Type	Fusion Function	Variable Inversion Functions	
		r_x	r_y
Int	$x + y$	$z - y$	$z - x$
	$x + c + y$	$z - c - y$	$z - c - x$
	$x * y$	$z\ div\ y$	$z\ div\ x$
	$c_1 * x + c_2 * y + c_3$	$(z - c_2 * y - c_3)\ div\ c_1$	$(z - c_1 * x - c_3)\ div\ c_2$
Real	$x + y$	$z - y$	$z - x$
	$x + c + y$	$z - c - y$	$z - c - x$
	$x * y$	z/y	z/x
	$c_1 * x + c_2 * y + c_3$	$(z - c_2 * y - c_3)/c_1$	$(z - c_1 * x - c_3)/c_2$
String	*x str++ y*	*str.substr z 0 (str.len x)*	*str.substr z (str.len x) (str.len y)*
	x str++ y	*str.substr z 0 (str.len x)*	*str.replace z x ""*
	x str++ c str++ y	*str.substr z 0 (str.len x)*	*str.replace (str.replace z x "") c ""*

报告提纲

从测试预言问题到蜕变测试

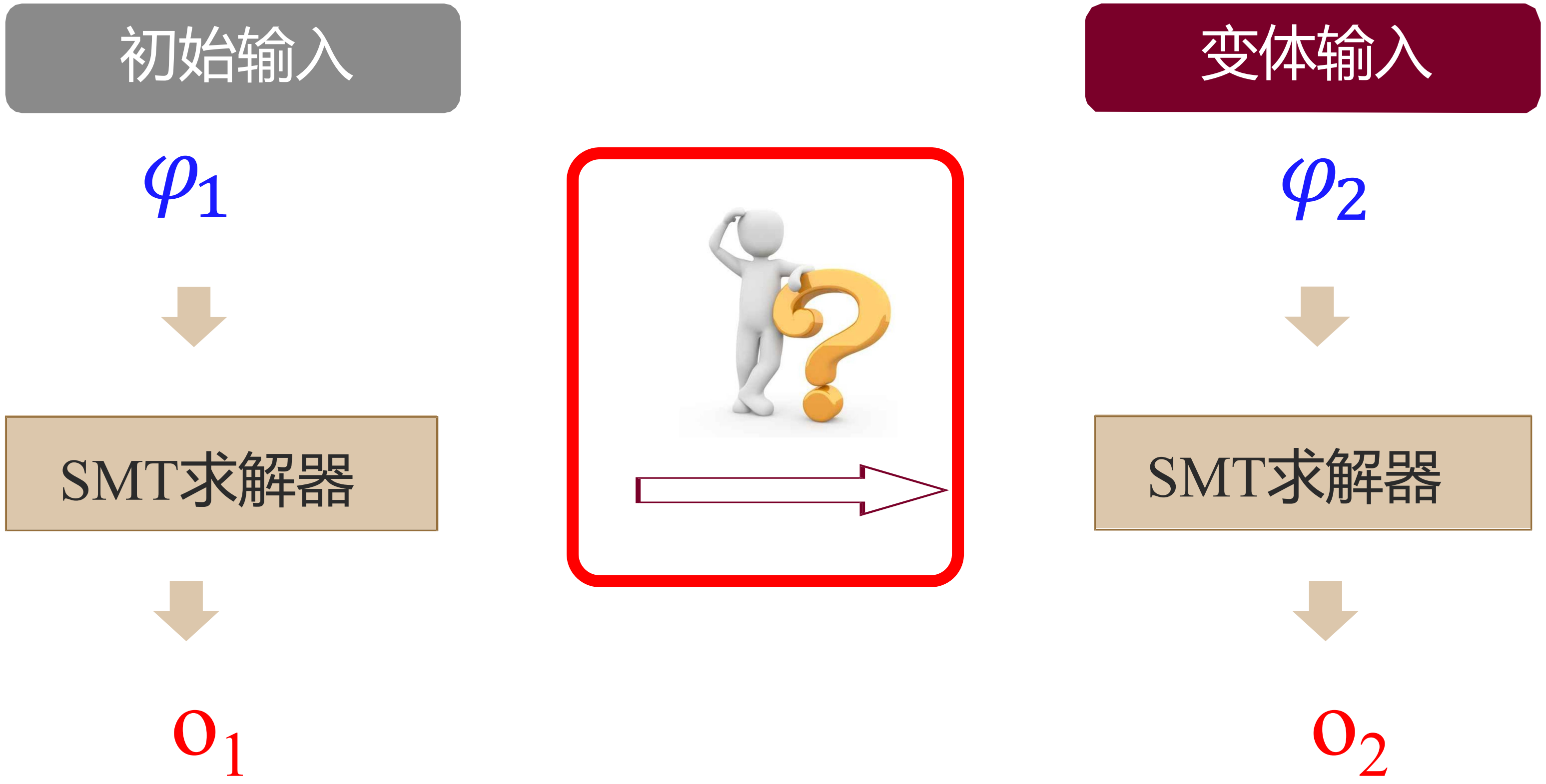
- 测试预言问题
- 蜕变测试概述
- 蜕变关系的属性

面向SMT求解器的蜕变测试

- SMT求解器及其逻辑缺陷
- 基于语义融合的方法
- **基于近似枚举的方法**

Sparrow-基于近似枚举的蜕变测试 [ESEC/FSE’21]

- 对**任意**种子约束做**等可满足性变换**



初始输入

φ_1

SMT求解器

o_1

变体输入

φ_2

SMT求解器

o_2

Skeletal Approximation Enumeration for SMT Solver Testing, ESEC/FSE’21

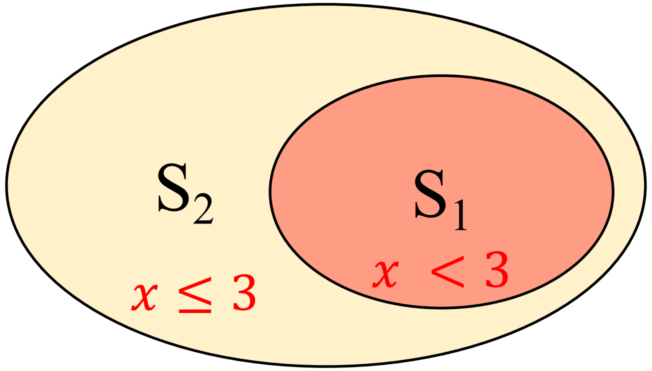
核心想法: 逻辑近似指导约束变换

下近似 $S_2 \rightarrow S_1$

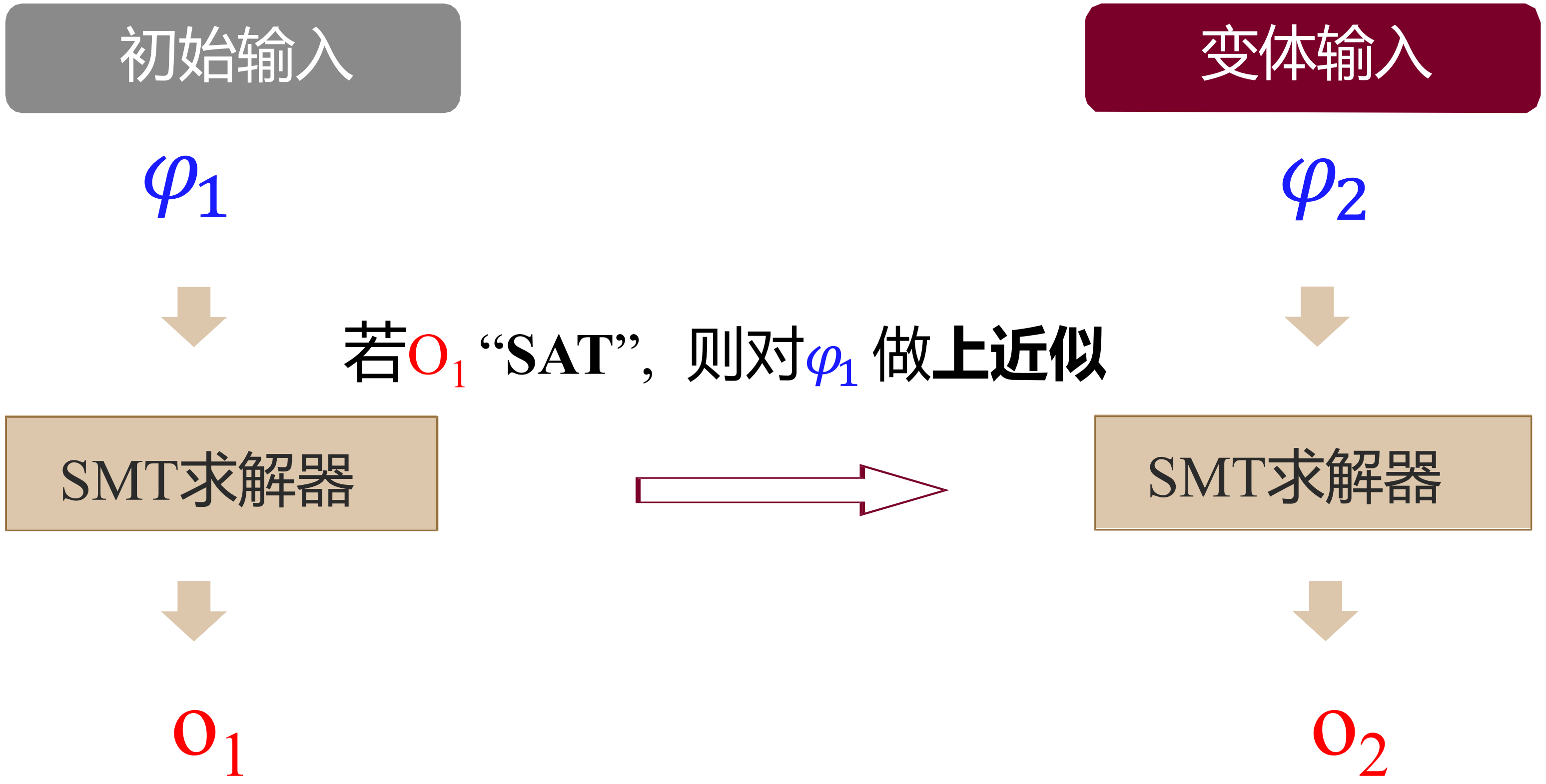
若S_2 UNSAT, 则S_1一定也UNSAT

上近似 $S_1 \rightarrow S_2$

若S_1 SAT, 则 S_2一定也SAT



逻辑近似指导的变换



初始输入

φ_1

SMT求解器

o_1

若o_1 “SAT”，则对φ_1 做**上近似**

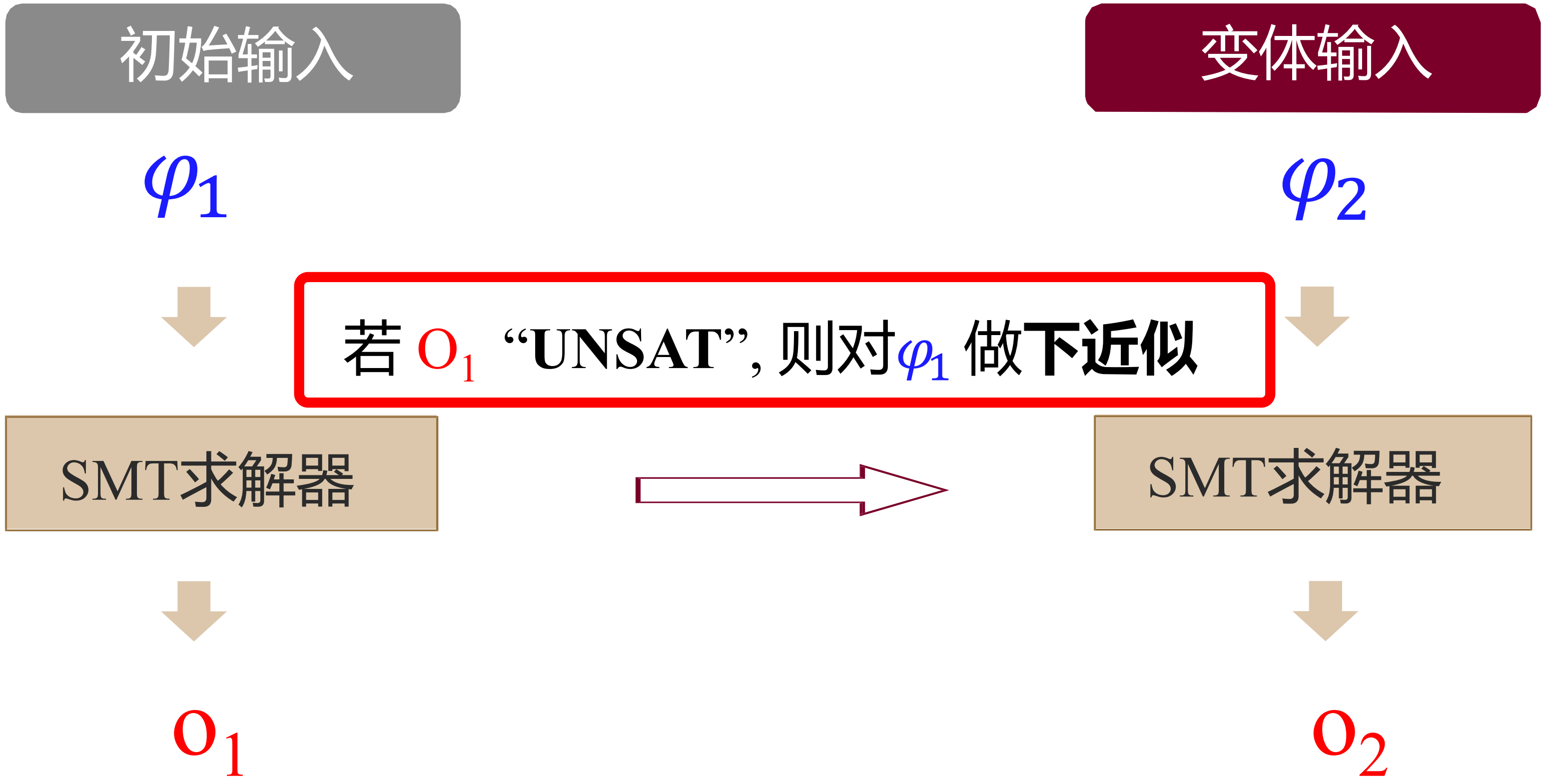
变体输入

φ_2

SMT求解器

o_2

逻辑近似指导的变换



初始输入

φ_1

SMT求解器

o_1

变体输入

φ_2

SMT求解器

o_2

若 o_1 “UNSAT”, 则对φ_1 做**下近似**

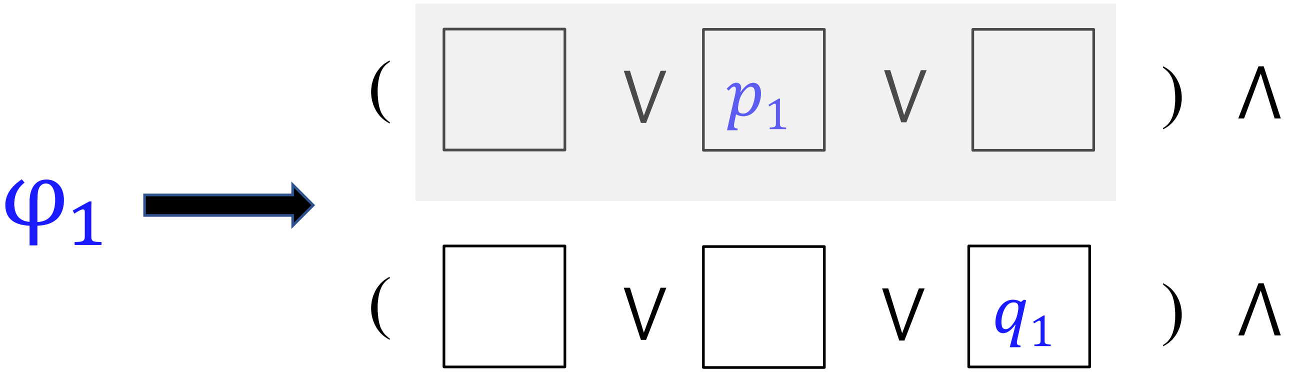
关键挑战: 如何对约束做上/下近似？

- 在形式化方法领域有广泛研究

- 大部分现有的算法
  - 只支持特定类型的约束, e.g., [Bryant et al. TACAS’07]
  - 涉及重量级的逻辑推理, e.g., [McMillan et al. CAV’06]

目标：**通用**且**轻量级**的逻辑近似方法?

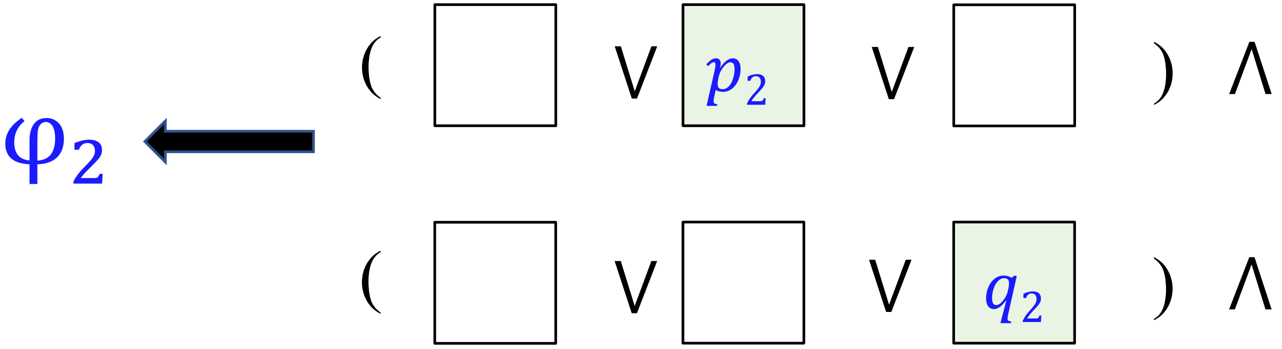
解决方案: 基于骨架的近似枚举



$$\varphi_1 \longrightarrow \begin{array}{l}(\;\square \lor \boxed{p_1} \lor \square\;)\;\land\\ (\;\square \lor \square \lor \boxed{q_1}\;)\;\land\end{array}$$

1. 变换到 Conjunctive Normal Form (CNF)形式

定理: 任意**局部**的上/下近似可以得到**全局**的上/下近似



$$\varphi_2 \longleftarrow \begin{array}{l}(\;\square \lor \boxed{p_2} \lor \square\;)\;\land\\ (\;\square \lor \square \lor \boxed{q_2}\;)\;\land\end{array}$$

2. 枚举局部/原子层面的近似

原子层面的近似策略

1. 谓词符号变换

- E.g., $x < y + 3$ 是对$x \leq y$ 的上近似(3为随机生成)

2. 约束片段植入

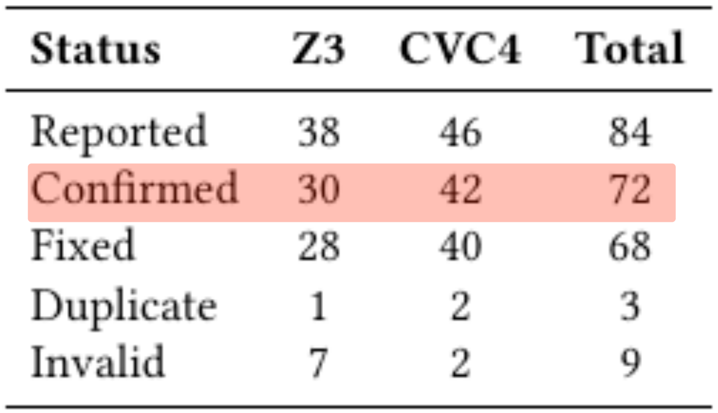
- “$p \lor f$” 是对p 的上近似
- “$p \land f$” 是对p 的下近似

f 是任意随机生成的约束片段

Sparrow部分实验结果

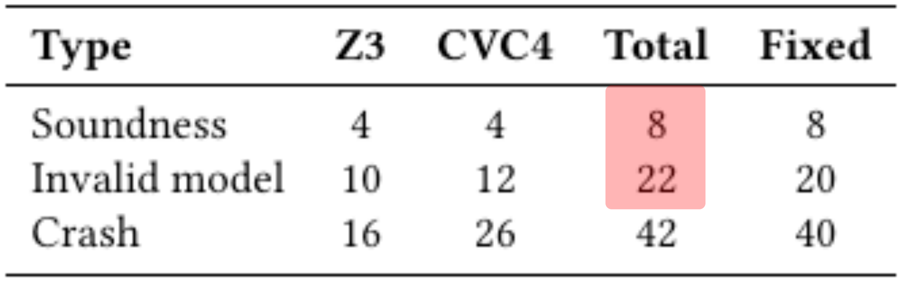
- 测试时间: 2020.11-2022.02

Table 3: Status of the bugs found by Sparrow.



Status	Z3	CVC4	Total
Reported	38	46	84
Confirmed	30	42	72
Fixed	28	40	68
Duplicate	1	2	3
Invalid	7	2	9

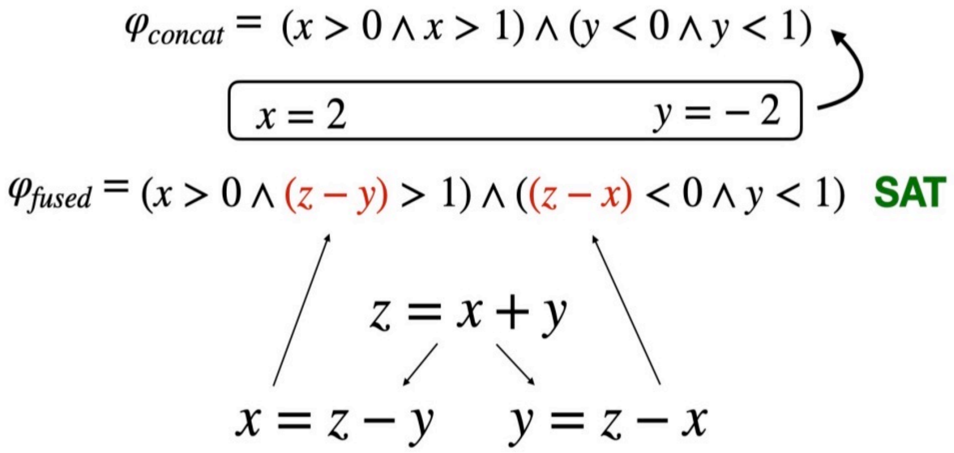
Table 4: Bug type of the confirmed bugs.



Type	Z3	CVC4	Total	Fixed
Soundness	4	4	8	8
Invalid model	10	12	22	20
Crash	16	26	42	40

小结: 面向SMT求解器的蜕变测试

基于**语义融合**的蜕变测试



$\varphi_{concat} = (x > 0 \wedge x > 1) \wedge (y < 0 \wedge y < 1)$

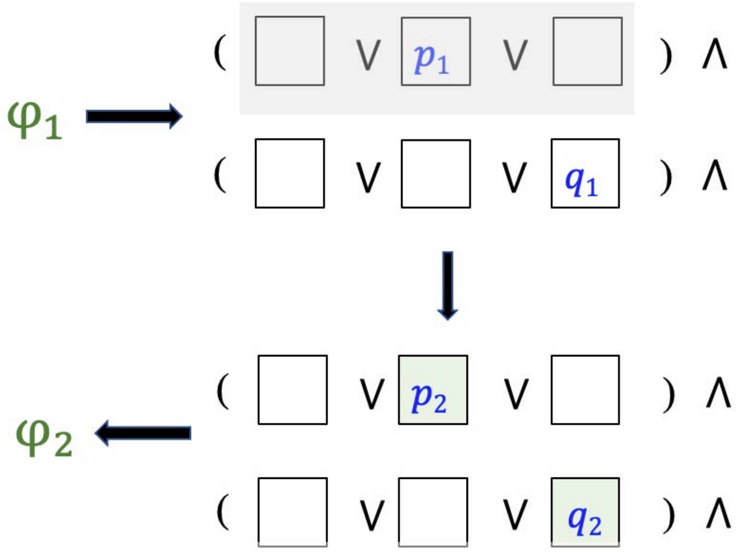
$\boxed{x = 2 \qquad y = -2}$

$\varphi_{fused} = (x > 0 \wedge (z - y) > 1) \wedge ((z - x) < 0 \wedge y < 1)$ SAT

$z = x + y$

$x = z - y \qquad y = z - x$

基于**近似枚举**的蜕变测试



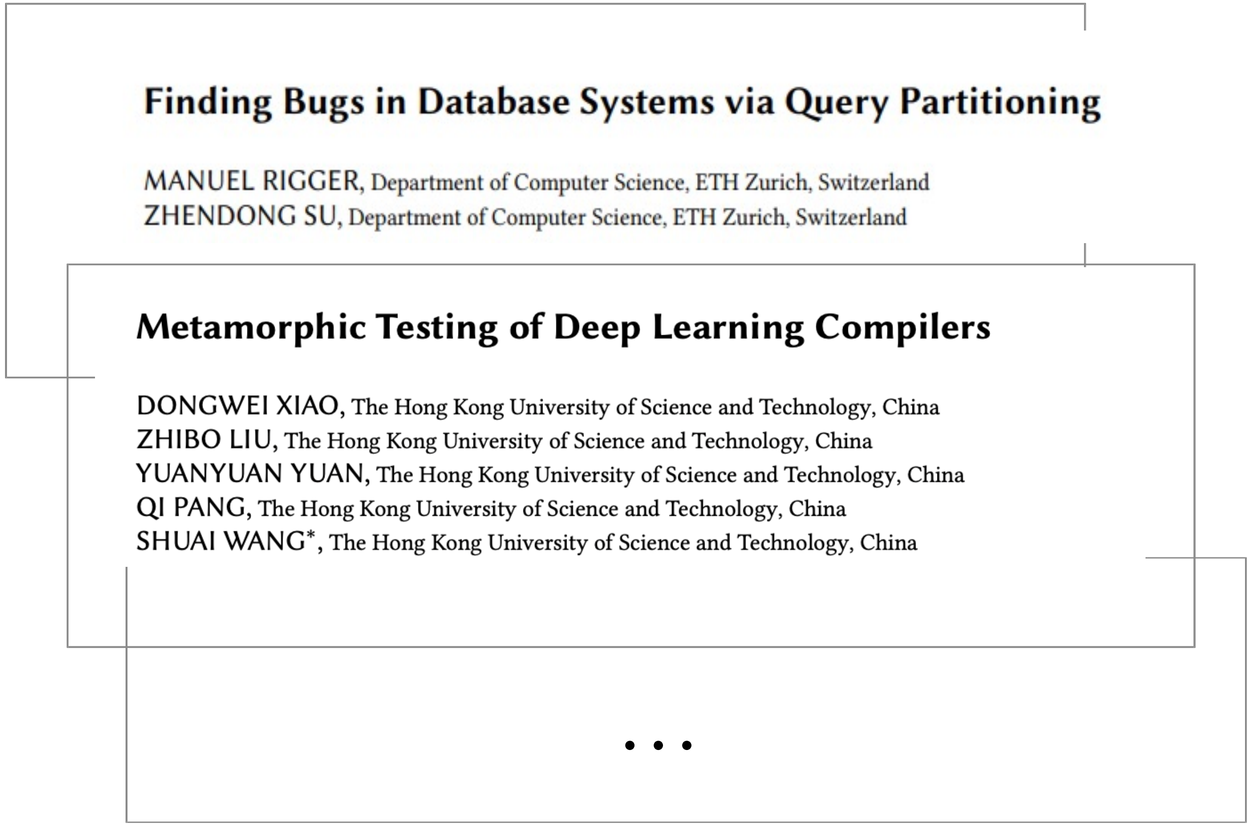
蜕变测试导引：课程小结

从测试预言问题到蜕变测试

面向SMT求解器的蜕变测试

开放问题: 蜕变关系在测试用例生成中的作用?

课后练习: 调研其它应用



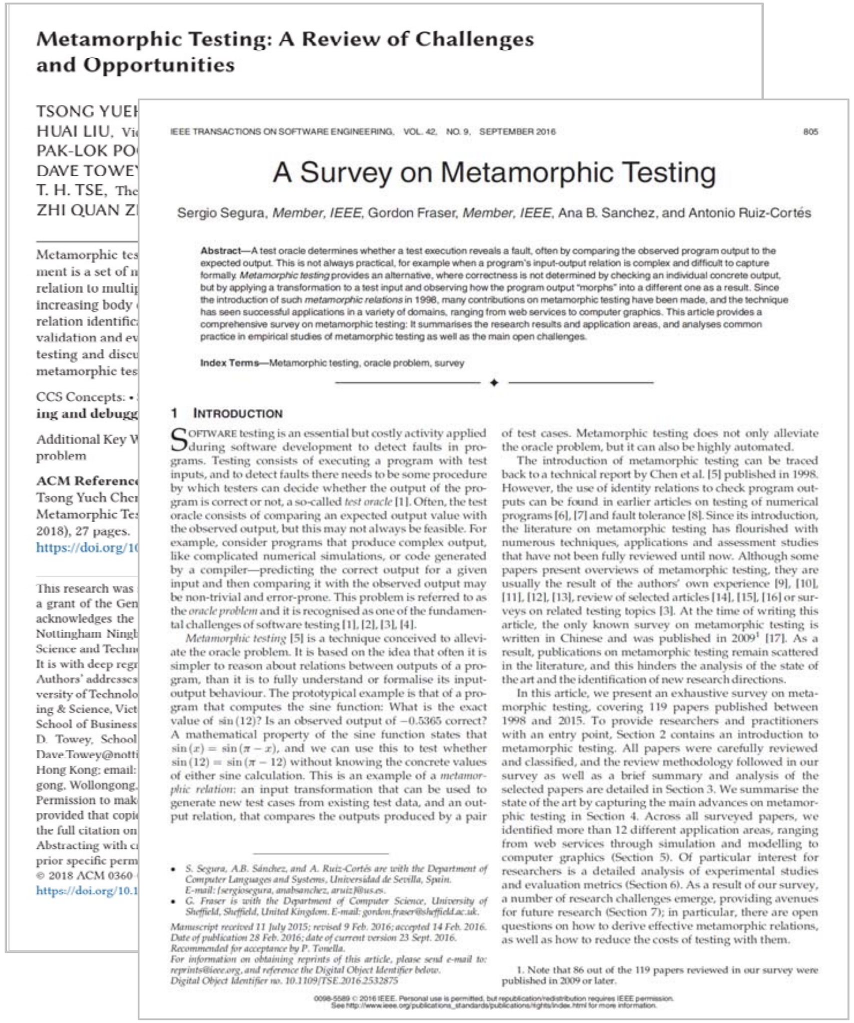
Finding Bugs in Database Systems via Query Partitioning

MANUEL RIGGER, Department of Computer Science, ETH Zurich, Switzerland
ZHENDONG SU, Department of Computer Science, ETH Zurich, Switzerland

Metamorphic Testing of Deep Learning Compilers

DONGWEI XIAO, The Hong Kong University of Science and Technology, China
ZHIBO LIU, The Hong Kong University of Science and Technology, China
YUANYUAN YUAN, The Hong Kong University of Science and Technology, China
QI PANG, The Hong Kong University of Science and Technology, China
SHUAI WANG*, The Hong Kong University of Science and Technology, China

• • •



Metamorphic Testing: A Review of Challenges and Opportunities

TSONG YUEI
HUAI LIU, Vi
PAK-LOK PO
DAVE TOWE
T. H. TSE, The
ZHI QUAN Z

Metamorphic tes
ment is a set of n
relation to multip
increasing body
relation identific
validation and ev
testing and discu
metamorphic tes

CCS Concepts: •
ing and debugg

Additional Key W
problem

ACM Referenc
Tsong Yuch Chen
Metamorphic Tes
2018), 27 pages.
https://doi.org/1

This research was
a grant of the Gen
acknowledges the
Nottingham Ningb
Science and Techn
It is with deep regr
Authors' addresses
versity of Technolo
ing & Science, Vict
School of Business
D. Towey, School
Dave.Towey@notti
Hong Kong; email:
gong, Wollongong.
Permission to mak
provided that copi
the full citation on
Abstracting with c
prior specific perm
© 2018 ACM 0360
https://doi.org/10.1

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 42, NO. 9, SEPTEMBER 2016 805

A Survey on Metamorphic Testing

Sergio Segura, *Member, IEEE*, Gordon Fraser, *Member, IEEE*, Ana B. Sanchez, and Antonio Ruiz-Cortés

Abstract—A test oracle determines whether a test execution reveals a fault, often by comparing the observed program output to the expected output. This is not always practical, for example when a program's input-output relation is complex and difficult to capture formally. *Metamorphic testing* provides an alternative, where correctness is not determined by checking an individual concrete output, but by applying a transformation to a test input and observing how the program output "morphs" into a different one as a result. Since the introduction of such *metamorphic relations* in 1998, many contributions on metamorphic testing have been made, and the technique has seen successful applications in a variety of domains, ranging from web services to computer graphics. This article provides a comprehensive survey on metamorphic testing: It summarises the research results and application areas, and analyses common practice in empirical studies of metamorphic testing as well as the main open challenges.

Index Terms—Metamorphic testing, oracle problem, survey

1 Introduction

SOFTWARE testing is an essential but costly activity applied during software development to detect faults in programs. Testing consists of executing a program with test inputs, and to detect faults there needs to be some procedure by which testers can decide whether the output of the program is correct or not, a so-called *test oracle* [1]. Often, the test oracle consists of comparing an expected output value with the observed output, but this may not always be feasible. For example, consider programs that produce complex output, like complicated numerical simulations, or code generated by a compiler—predicting the correct output for a given input and then comparing it with the observed output may be non-trivial and error-prone. This problem is referred to as the *oracle problem* and it is recognised as one of the fundamental challenges of software testing [1], [2], [3], [4].

Metamorphic testing [5] is a technique conceived to alleviate the oracle problem. It is based on the idea that often it is simpler to reason about relations between outputs of a program, than it is to fully understand or formalise its input-output behaviour. The prototypical example is that of a program that computes the sine function: What is the exact value of $\sin(12)$? Is an observed output of -0.5365 correct? A mathematical property of the sine function states that $\sin(x) = \sin(\pi - x)$, and we can use this to test whether $\sin(12) = \sin(\pi - 12)$ without knowing the concrete values of either sine calculation. This is an example of a *metamorphic relation*: an input transformation that can be used to generate new test cases from existing test data, and an output relation, that compares the outputs produced by a pair of test cases. Metamorphic testing does not only alleviate the oracle problem, but it can also be highly automated.

The introduction of metamorphic testing can be traced back to a technical report by Chen et al. [5] published in 1998. However, the use of identity relations to check program outputs can be found in earlier articles on testing of numerical programs [6], [7] and fault tolerance [8]. Since its introduction, the literature on metamorphic testing has flourished with numerous techniques, applications and assessment studies that have not been fully reviewed until now. Although some papers present overviews of metamorphic testing, they are usually the result of the authors' own experience [9], [10], [11], [12], [13], review of selected articles [14], [15], [16] or surveys on related testing topics [3]. At the time of writing this article, the only known survey on metamorphic testing is written in Chinese and was published in 2009[1] [17]. As a result, publications on metamorphic testing remain scattered in the literature, and this hinders the analysis of the state of the art and the identification of new research directions.

In this article, we present an exhaustive survey on metamorphic testing, covering 119 papers published between 1998 and 2015. To provide researchers and practitioners with an entry point, Section 2 contains an introduction to metamorphic testing. All papers were carefully reviewed and classified, and the review methodology followed in our survey as well as a brief summary and analysis of the selected papers are detailed in Section 3. We summarise the state of the art by capturing the main advances on metamorphic testing in Section 4. Across all surveyed papers, we identified more than 12 different application areas, ranging from web services through simulation and modelling to computer graphics (Section 5). Of particular interest for researchers is a detailed analysis of experimental studies and evaluation metrics (Section 6). As a result of our survey, a number of research challenges emerge, providing avenues for future research (Section 7); in particular, there are open questions on how to derive effective metamorphic relations, as well as how to reduce the costs of testing with them.

- *S. Segura, A.B. Sánchez, and A. Ruiz-Cortés are with the Department of Computer Languages and Systems, Universidad de Sevilla, Spain. E-mail: {sergiosegura, anabsanchez, aruiz}@us.es.*
- *G. Fraser is with the Department of Computer Science, University of Sheffield, Sheffield, United Kingdom. E-mail: gordon.fraser@sheffield.ac.uk.*

Manuscript received 11 July 2015; revised 9 Feb. 2016; accepted 14 Feb. 2016. Date of publication 28 Feb. 2016; date of current version 23 Sept. 2016. Recommended for acceptance by P. Tonella. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TSE.2016.2532875

[1] Note that 86 out of the 119 papers reviewed in our survey were published in 2009 or later.

0098-5589 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

欢迎深造/报考！

港科大

浙大