

智能合约动态验证

Securing Smart Contract with Runtime Validation

Ao Li, Jemin Andrew Choi, Fan Long - University of Toronto

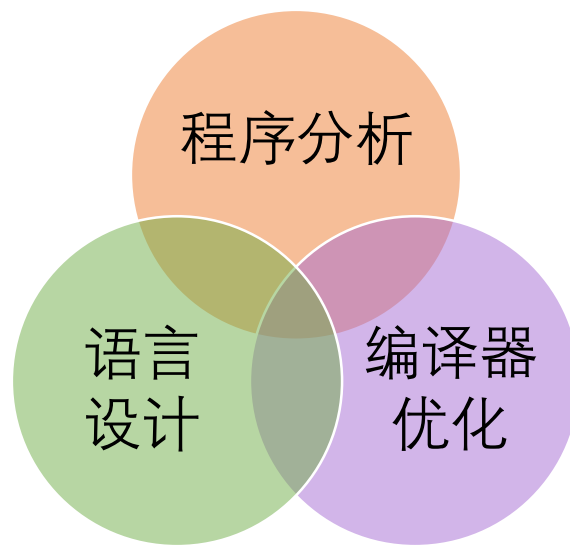
王子彦

ziyan-wang.github.io

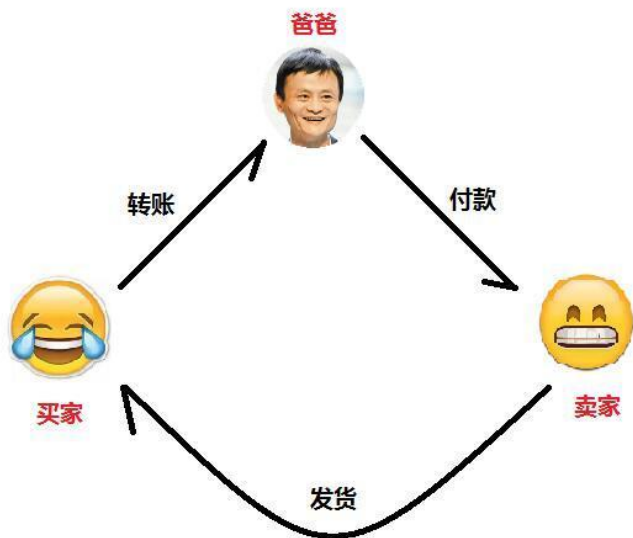
2021/06/24

王子彦

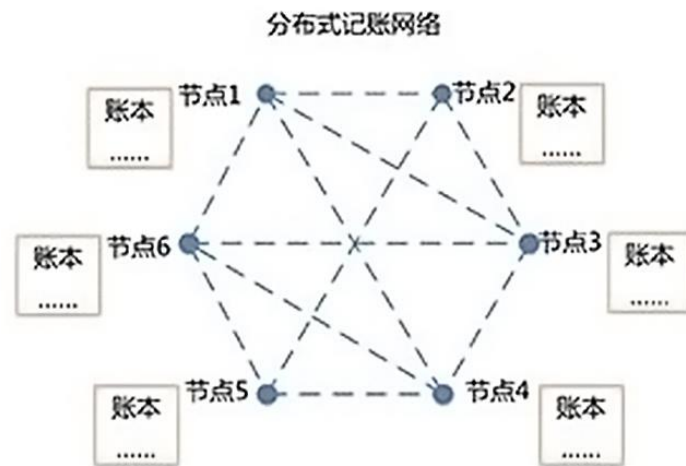
- 研一@InPlusLab



区块链



中心化交易

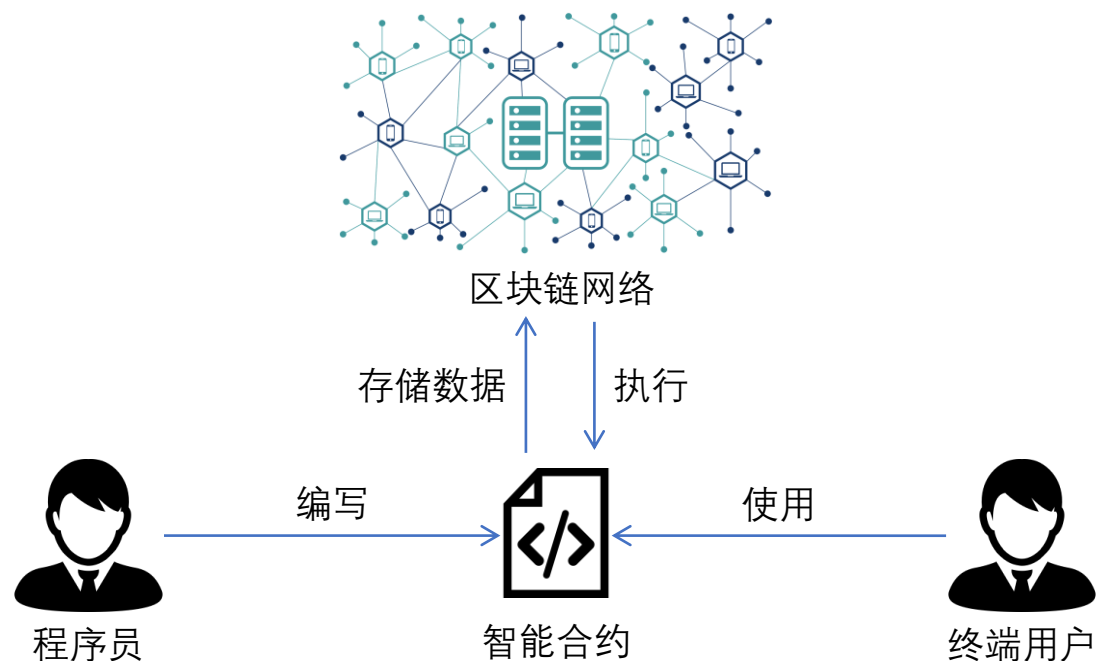


分布式账本

去中心化交易（区块链）

智能合约

- 智能合约是在区块链上执行的代码
- 相同的代码会在区块链网络的所有节点（机器）上几乎同时执行
- 用户在使用智能合约时，需要支付费用



智能合约的例子：投票

```
1  contract Vote {
2      mapping(int => int) public voteCounts;
3
4      function vote(int option) public {
5          voteCounts[option] += 1;
6      }
7  }
```

Solidity 语言
(智能合约)

```
1  class Vote {
2      public Map<Integer, Integer> voteCounts = new HashMap<>();
3
4      public void vote(int option) {
5          if (!voteCounts.containsKey(option)) {
6              voteCounts.put(option, 0);
7          }
8          voteCounts.put(option, voteCounts.get(option) + 1);
9      }
10 }
```

Java

```
1  class Vote:
2      def __init__(self):
3          self.voteCounts = {}
4
5      def vote(self, option):
6          if option not in self.voteCounts.keys():
7              self.voteCounts[option] = 0
8              self.voteCounts[option] += 1
```

Python

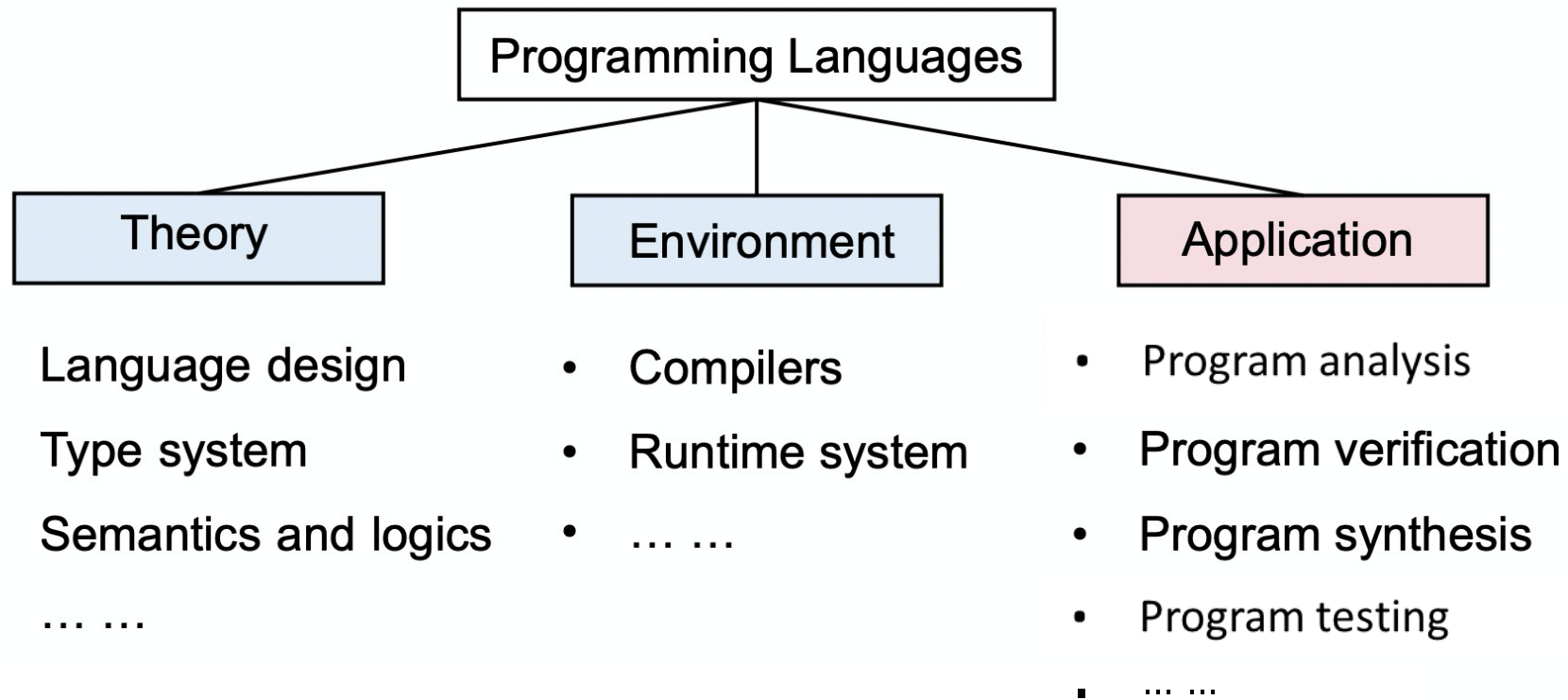
Problem

- 智能合约的代码是人写的，会有bug
- 智能合约的bug可导致严重后果，因为
 - 以太坊上的合约一旦部署，就不可再直接修改其代码
 - 合约内部的bug被视为正常的预期行为，在区块链上忠实执行（不可逆）
 - 合约存储和管理着数字资产和数字身份
- 通常导致大量的经济损失
 - 2016年DAO攻击事件中，黑客窃取超过5000万美元

Problem

- 智能合约的代码是人写的，会有bug
- 智能合约的bug可导致严重后果，因为
 - 以太坊上的合约一旦部署，就不可再直接修改其代码
 - 合约内部的bug被视为正常的预期行为，在区块链上忠实执行（不可逆）
 - 合约存储和管理着数字资产和数字身份
- 通常导致大量的经济损失
 - 2016年DAO攻击事件中，黑客窃取超过5000万美元

如何减少bug?





60年代干打垒
面向语句



70年代混合结构
面向功能模块



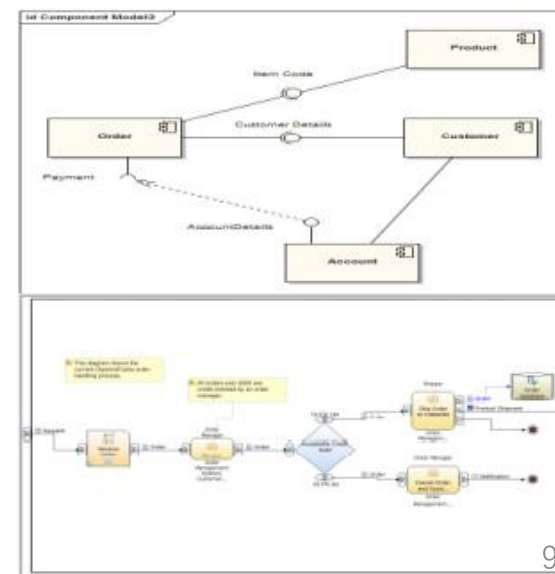
80年代框架结构
面向对象

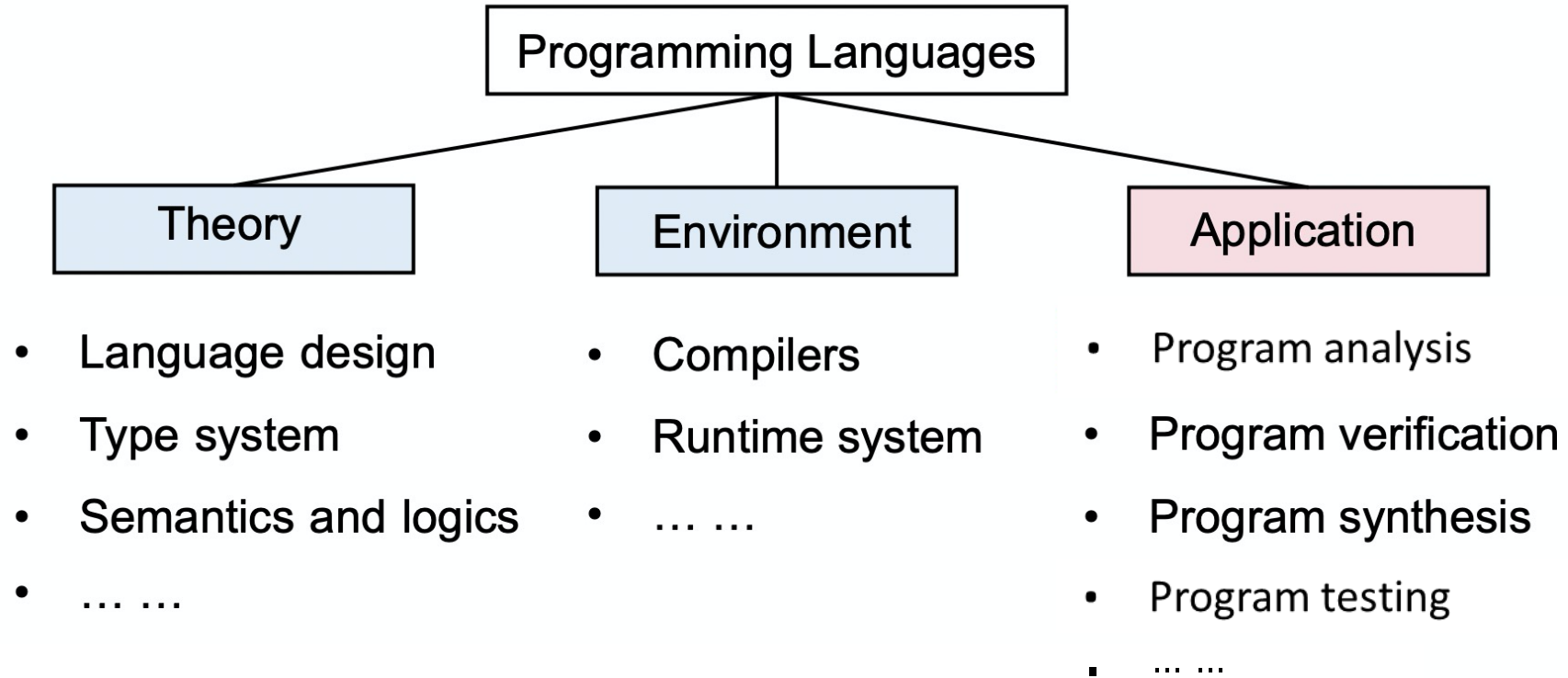


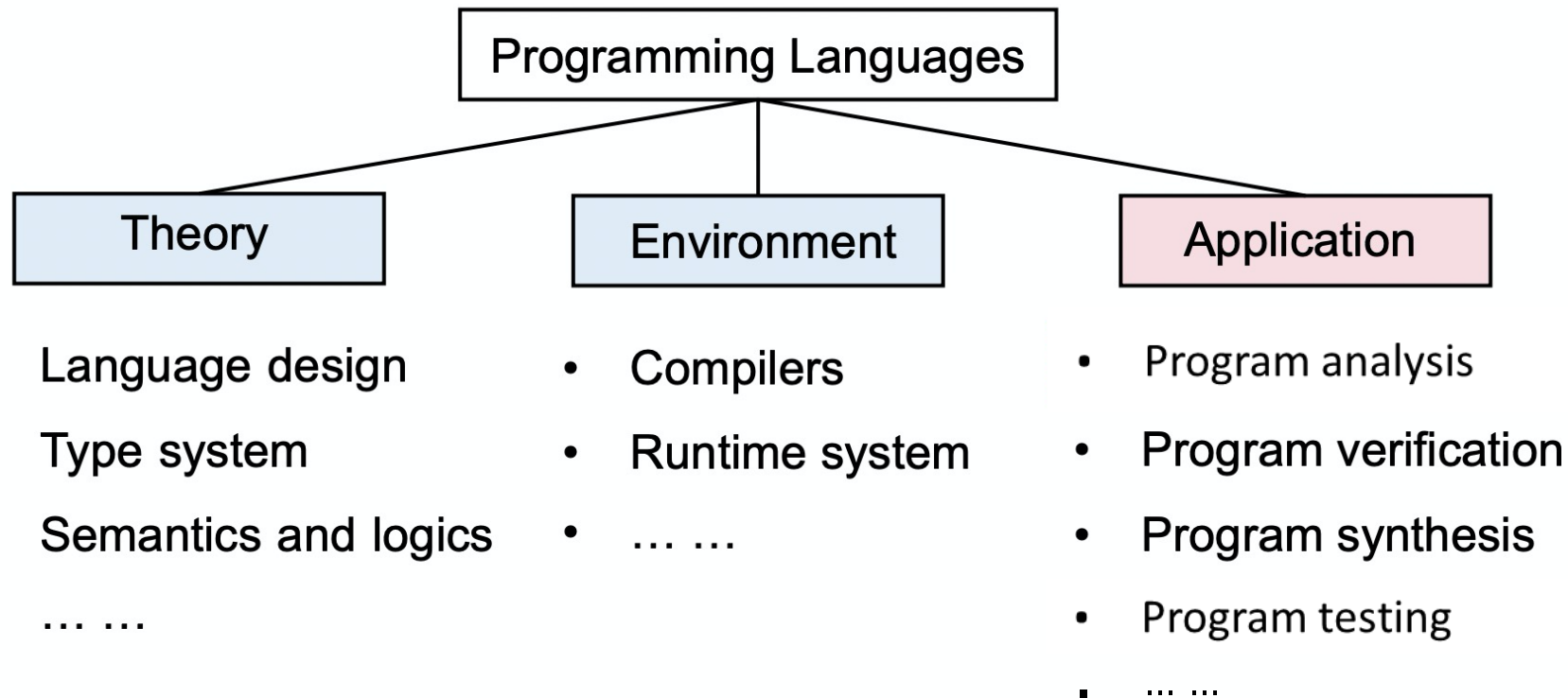
90年代钢结构
面向组件、业务

```

ORAL, 30H
CALLDISP
MOVVAL, BL
ANDAL, BFH
ORAL, 30H
CALLDISP
DECCX
JCXZDONE
MOVVAL, ,
CALLDISP
JMPNEXT
DONE, MOVVAL, 4CH;终止当前进程,返回调用进程
INT21H
DISP, MOVVAL, 14 ;写字符到当前光标位置,光标前进一格
INT10H
RET
CSEGENDS
ENDSTART
    
```

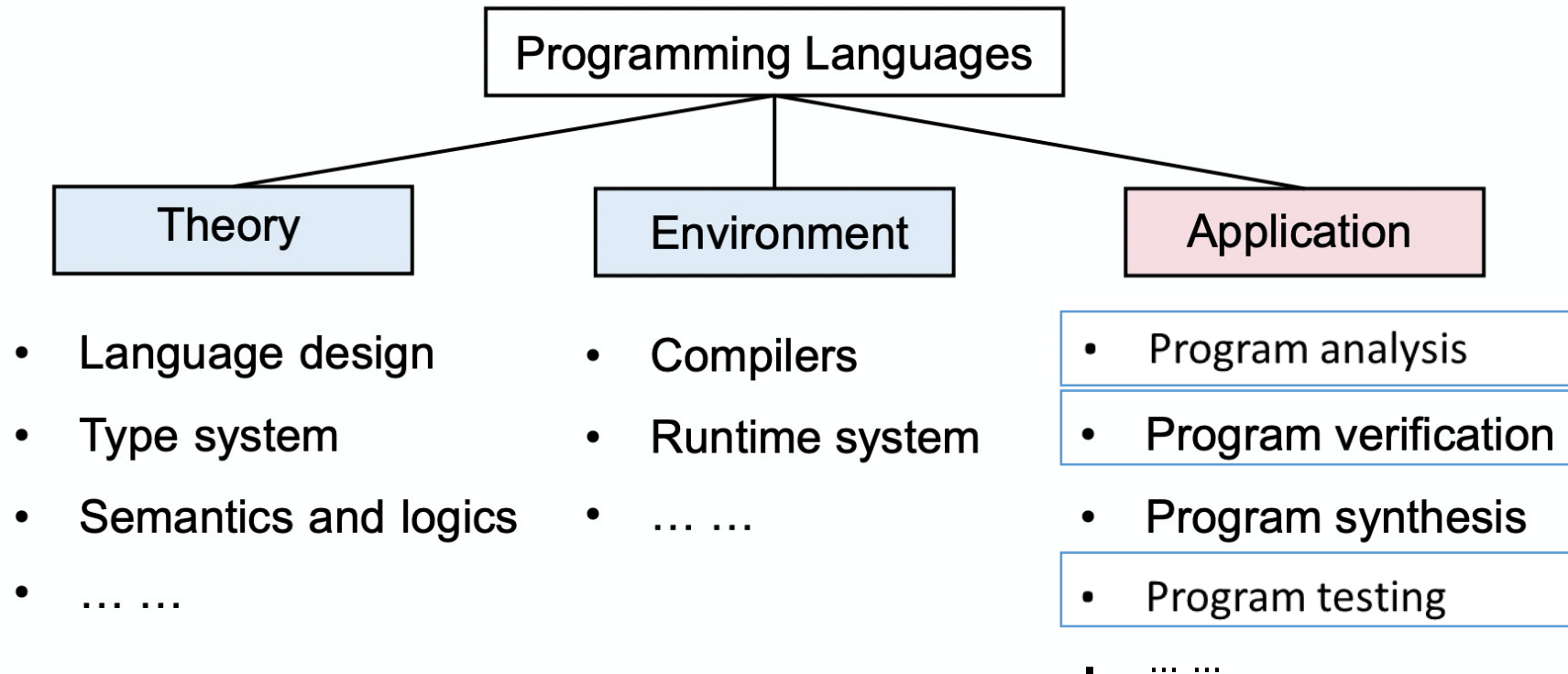






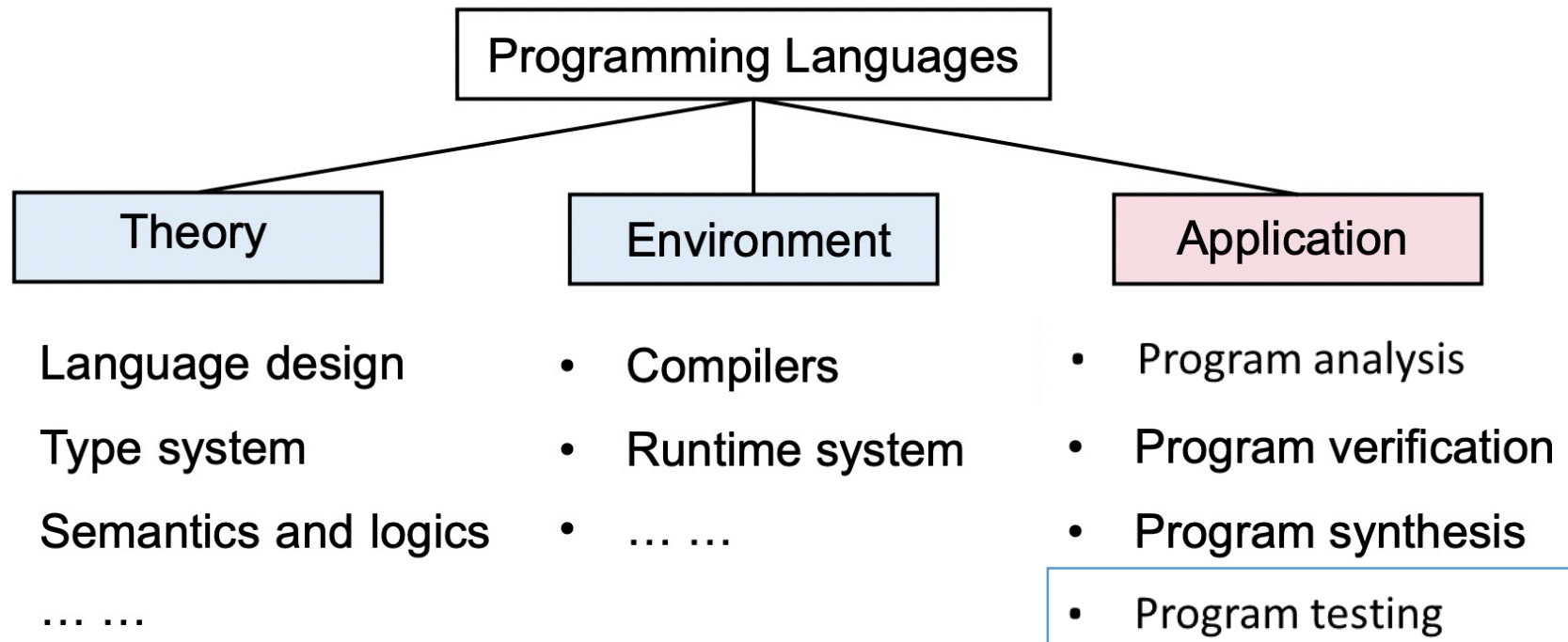
Background: In the last decade, the language cores had few changes, but the programs became significantly larger and more complicated.

Challenge: How to ensure the reliability, security and other promises of large-scale and complex programs?



Background: In the last decade, the language cores had few changes, but the programs became significantly larger and more complicated.

Challenge: How to ensure the reliability, security and other promises of large-scale and complex programs?



Background: In the last decade, the language cores had few changes, but the programs became significantly larger and more complicated.

Challenge: How to ensure the reliability, security and other promises of large-scale and complex programs?

vuejs / vue

Sponsor

Watch

6.3k

Star

178k

Fork

27.9k

Code

Issues 344

Pull requests 216

Actions

Projects 3

Wiki

Security

...

dev

Go to file

Code

About

Vue.js is a progressive, incrementally-adoptable JavaScript framework for building UI on the web.

vuejs.org

javascript

framework

vue

frontend



vue-bot chore: update sponsors [ci skip] (#11866)

6 days ago 3,136



src

build: unexpected token when running dev (#11704)

4 months ago



test

fix(v-pre): do not alter attributes (#10088)

4 months ago



types

fix(types): allow string for watch handlers in options...

4 months ago



.babelrc.js

workflow: upgraded to babel 7 (#8948)

2 years ago

单元测试

```
1  contract Demo {  
2      int public c;  
3      function add(int a, int b) public {  
4          c = a + b;  
5      }  
6  }  
7  
8  contract DemoTest {  
9      Demo demo;  
10     function testAdd() public {  
11         demo.add(1, 2);  
12         assert(demo.c == 1 + 2);  
13     }  
14 }
```

主程序


测试程序

测试的局限性

- 1972 年的图灵奖得主 Edsger Wybe Dijkstra 说道: “Program testing can be used to show the presence of bugs, but never to show their absence! ”, 即测试只能表明程序中存在错误, 而不能表明程序中没有错误。除非对程序进行的测试能够穷尽所有可能的场景, 否则传统的测试手段无法完全保证系统的安全可靠。


测试的局限性


- 1972 年的图灵奖得主 Edsger Wybe Dijkstra 说道: “Program testing can be used to show the presence of bugs, but never to show their absence! ”, 即测试只能表明程序中存在错误, 而不能表明程序中没有错误。除非对程序进行的测试能够穷尽所有可能的场景, 否则传统的测试手段无法完全保证系统的安全可靠。

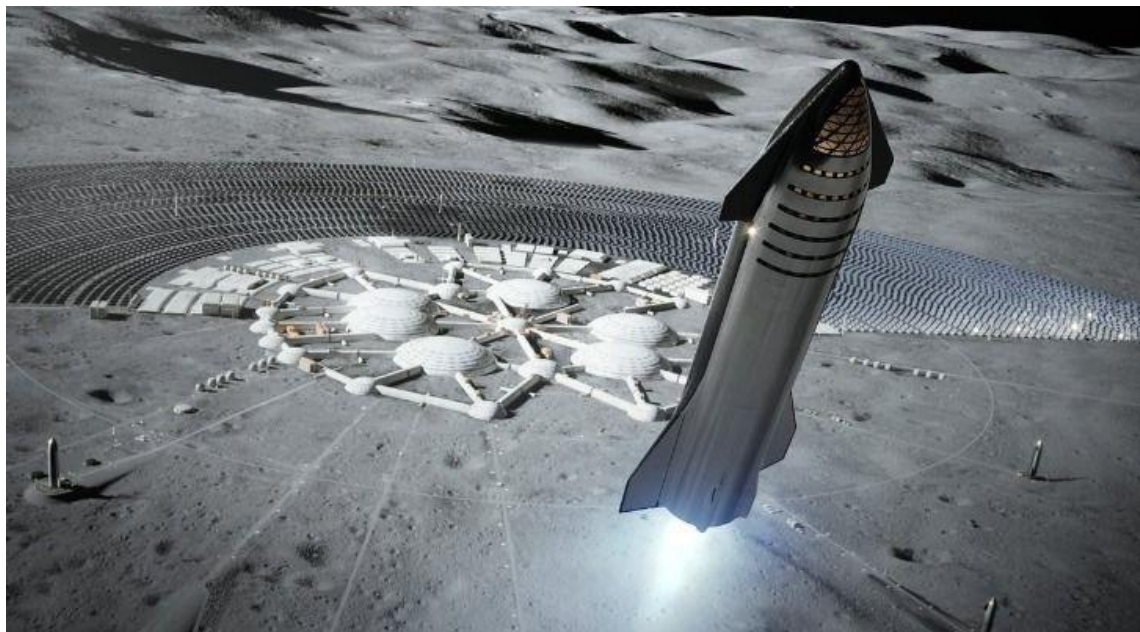
int a;  [-2147483648, 2147483647] 42亿个数

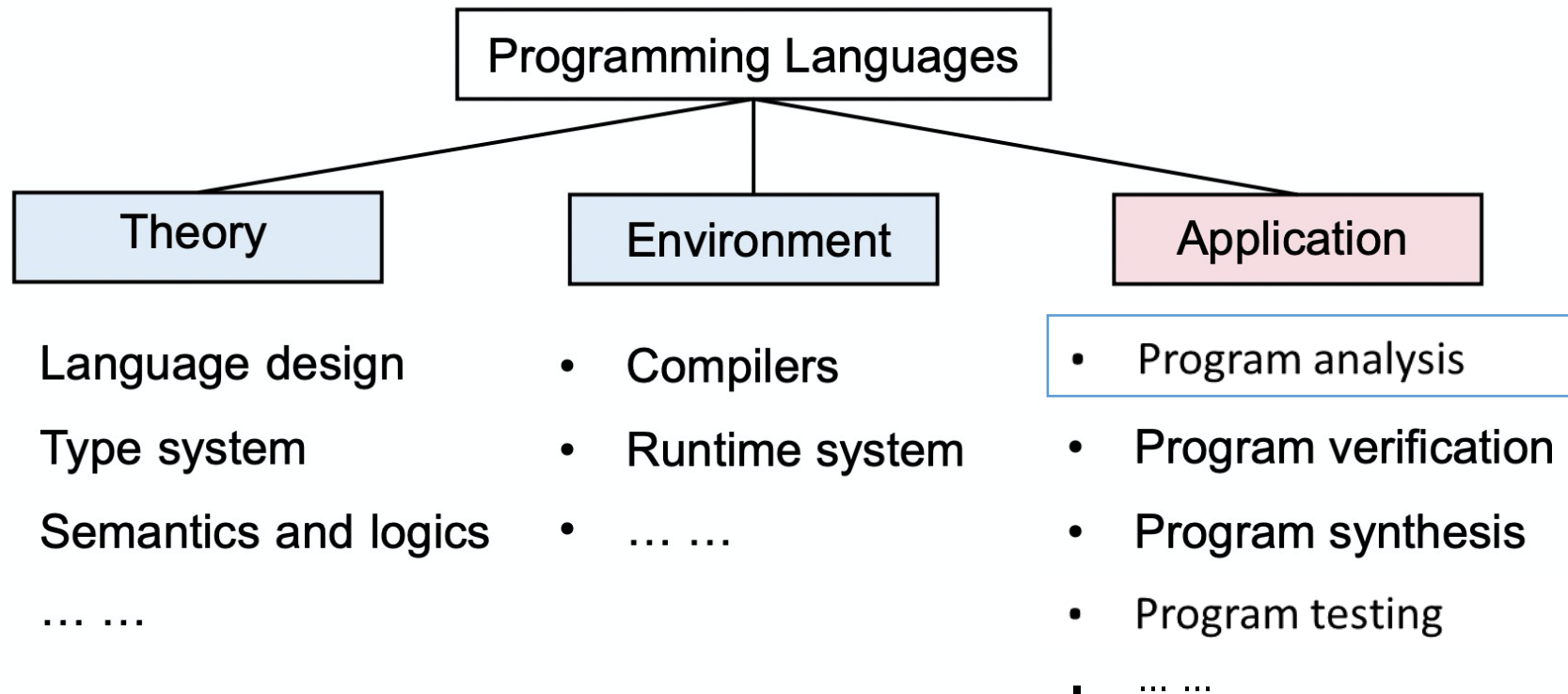
测试的局限性

- 1972 年的图灵奖得主 Edsger Wybe Dijkstra 说道: “Program testing can be used to show the presence of bugs, but never to show their absence!”, 即测试只能表明程序中存在错误, 而不能表明程序中没有错误。除非对程序进行的测试能够穷尽所有可能的场景, 否则传统的测试手段无法完全保证系统的安全可靠。

int a;  [-2147483648, 2147483647] 42亿个数

int a, b;  (42亿×42亿)个数





Background: In the last decade, the language cores had few changes, but the programs became significantly larger and more complicated.

Challenge: How to ensure the reliability, security and other promises of large-scale and complex programs?

静态分析可以回答的问题

- 程序会发生空指针异常吗？
- 程序中的类型转换都正确吗？
- 程序中的assert语句会执行失败吗？
- ...

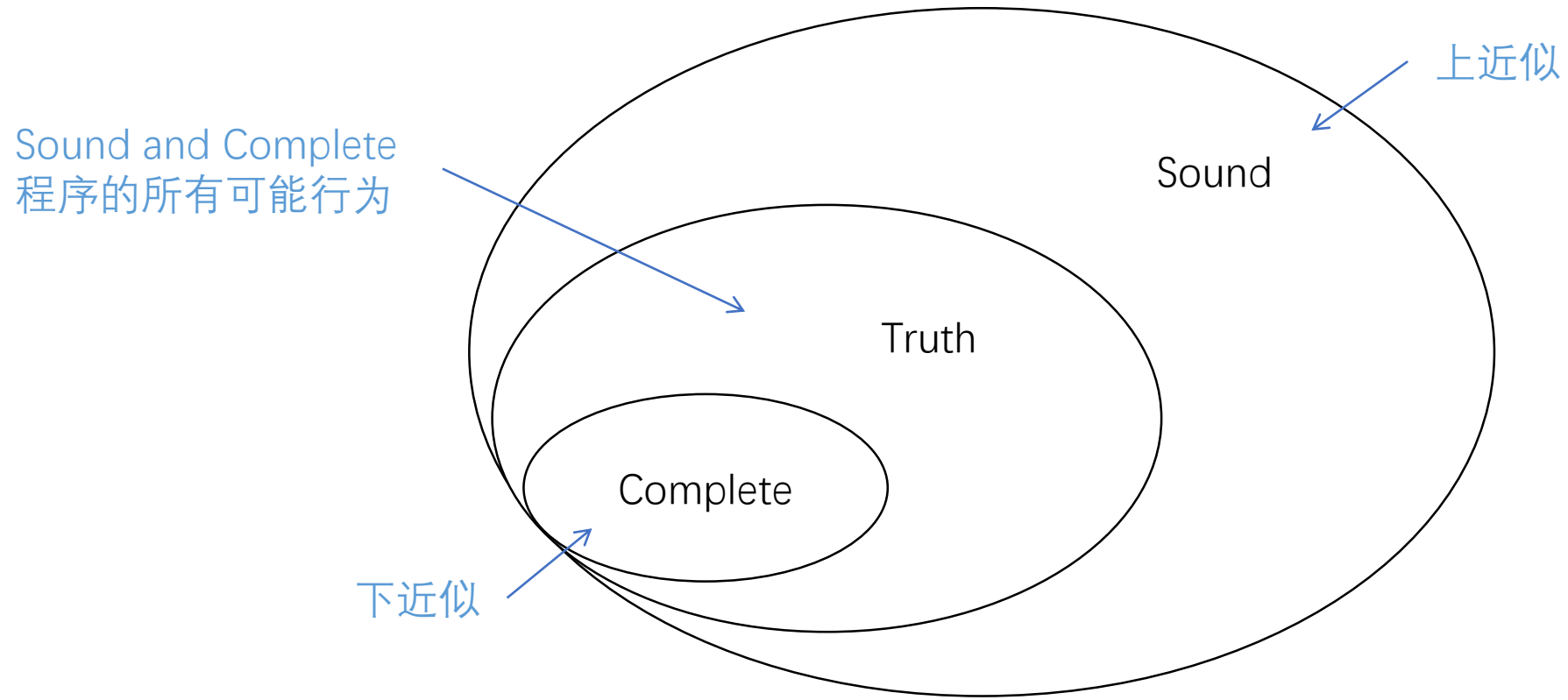
静态分析可以回答的问题

- 程序会发生空指针异常吗？
- 程序中的类型转换都正确吗？
- 程序中的assert语句会执行失败吗？

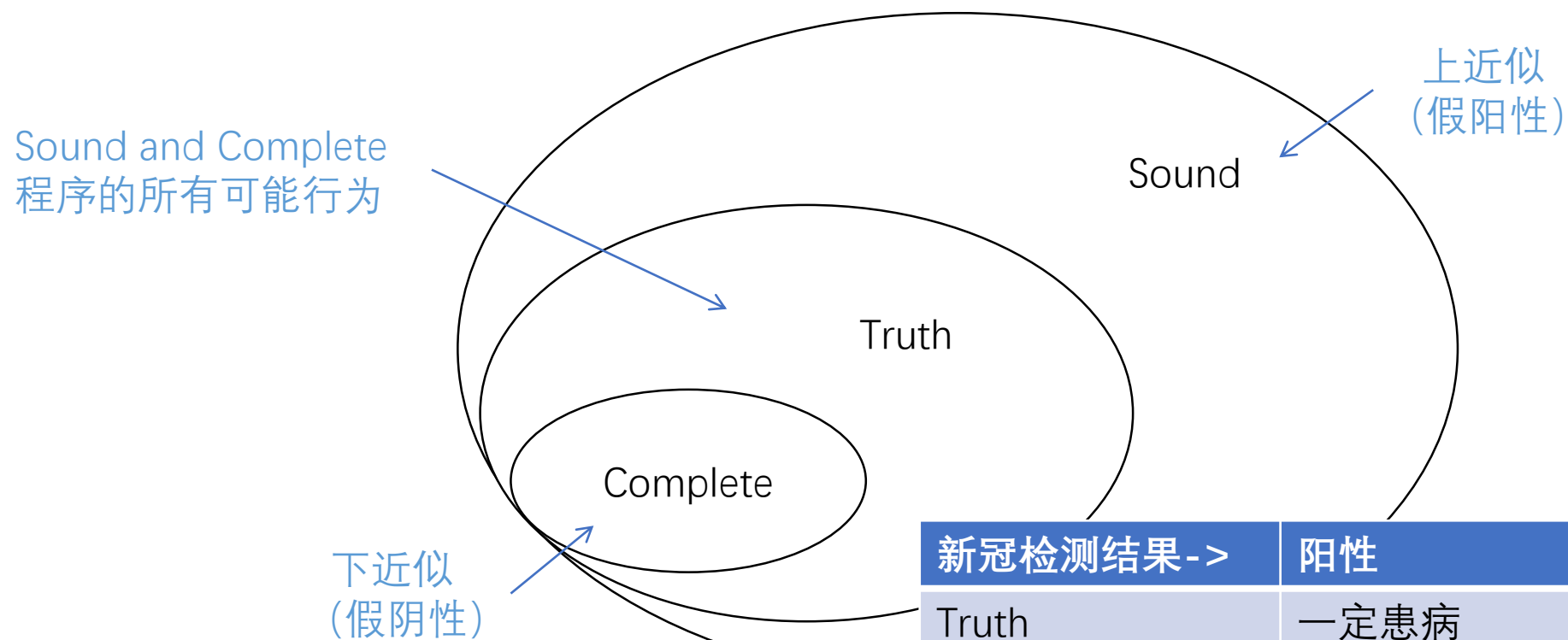
无法保证
回答正确

- 完美的分析是不可能的，必然存在假阳性或假阴性

Sound / Complete



Sound / Complete

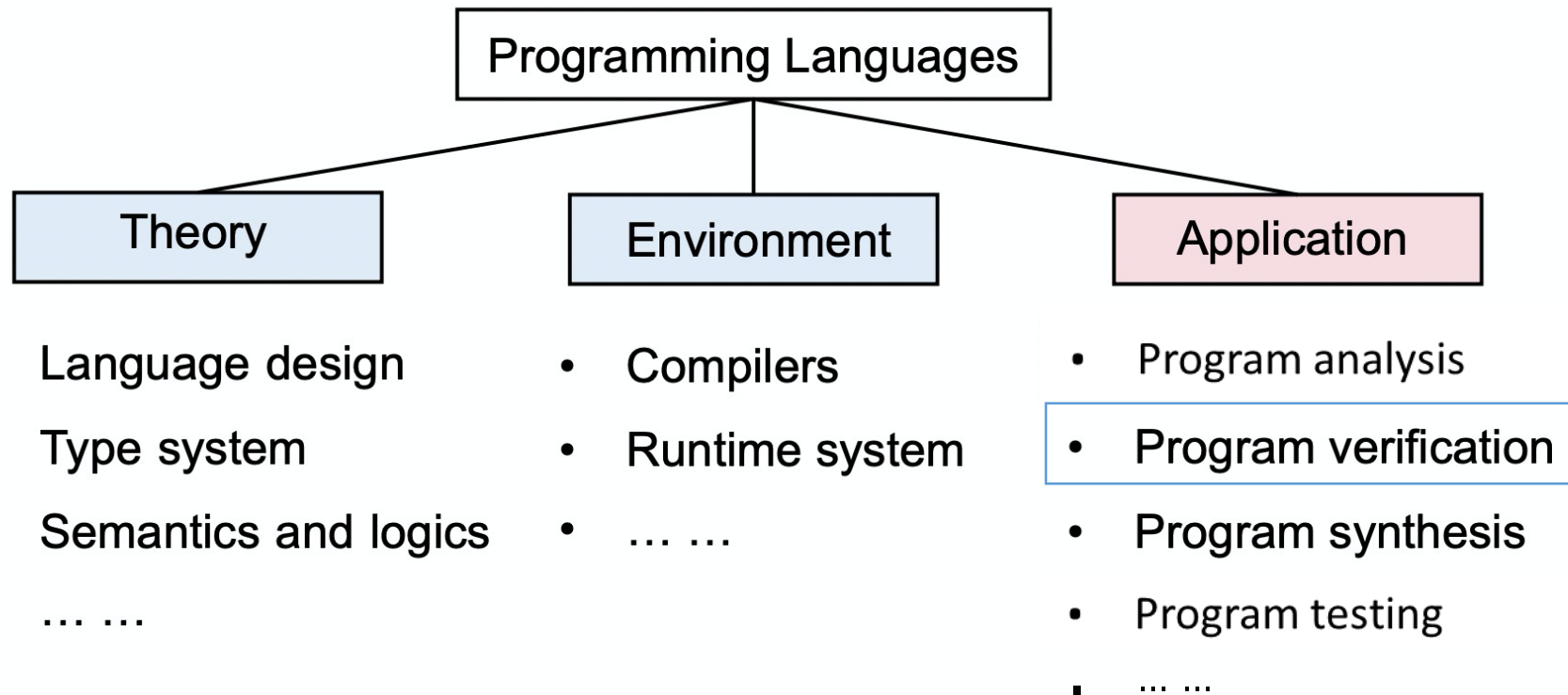


举例

新冠检测结果 ->	阳性	阴性
Truth	一定患病	一定健康
Sound	可能患病或健康	一定健康
Complete	一定患病	可能患病或健康

常用的静态分析技术

- **词法分析**：从左至右一个字符一个字符的读入源程序，对构成源程序的字符流进行扫描，通过使用正则表达式匹配方法将源代码转换为等价的符号（Token）流，生成相关符号列表。
- **语法分析**：判断源程序结构上是否正确，通过使用上下文无关语法将相关符号整理为语法树。
- **抽象语法树分析**（AST）：将程序组织成树形结构，树中相关节点代表了程序中的相关代码。
- **语义分析**：对结构上正确的源程序进行上下文有关性质的审查。
- **控制流分析**：生成有向控制流图，用节点表示基本代码块，节点间的有向边代表控制流路径，反向边表示可能存在的循环；还可生成函数调用关系图，表示函数间的嵌套关系。
- **数据流分析**：对控制流图进行遍历，记录变量的初始化点和引用点，保存切片相关数据信息。
- **污点分析**：基于数据流图，判断源代码中哪些变量可能受到攻击，是验证程序输入、识别代码表达缺陷的关键。
- **无效代码分析**：根据控制流图可分析孤立的节点部分为无效代码。



Background: In the last decade, the language cores had few changes, but the programs became significantly larger and more complicated.


Challenge: How to ensure the reliability, security and other promises of large-scale and complex programs?

程序验证


- 传统方法：形式化方法
 - 将程序抽象成一个数学公式，用严密的数学推理来证明（证实或证伪）该公式
- 一种形式化方法：模型检查
 - 判定代码是否满足规约（Specification）或不变式（Invariant）

程序验证

- 传统方法：形式化方法
 - 将程序抽象成一个数学公式，用严密的数学推理来证明（证实或证伪）该公式
- 一种形式化方法：模型检查
 - 判定代码是否满足规约（Specification）或不变式（Invariant）

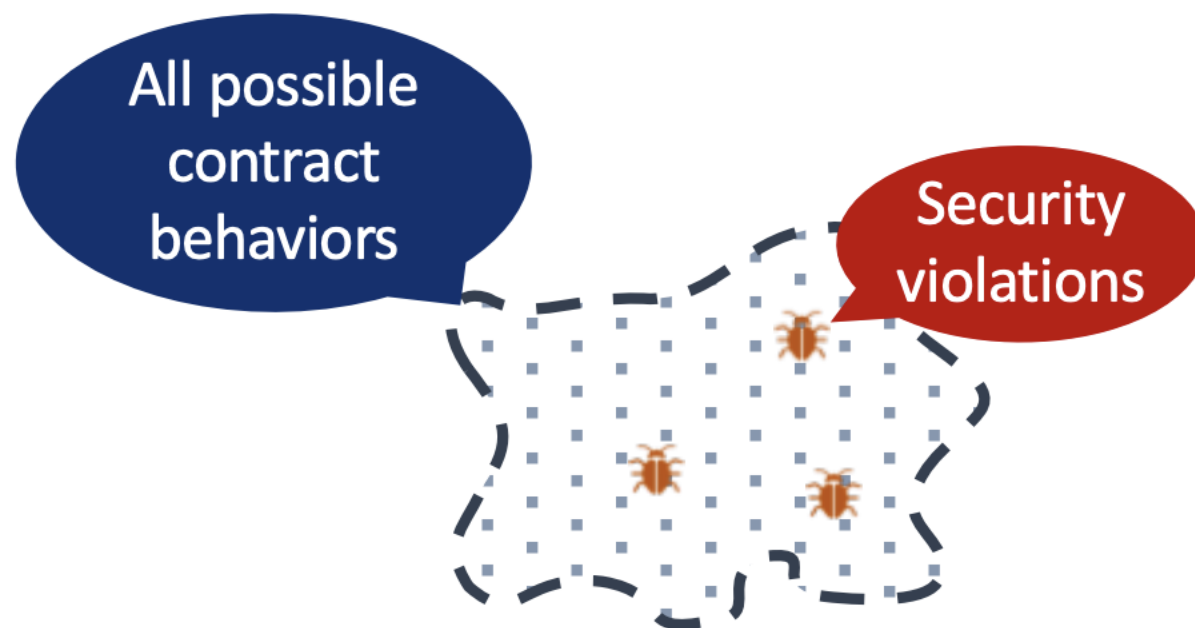


```
1  contract InvariantDemo {  
2      uint c;  
3      function add(uint a, uint b) {  
4          c = a + b;  
5      }  
6  }
```

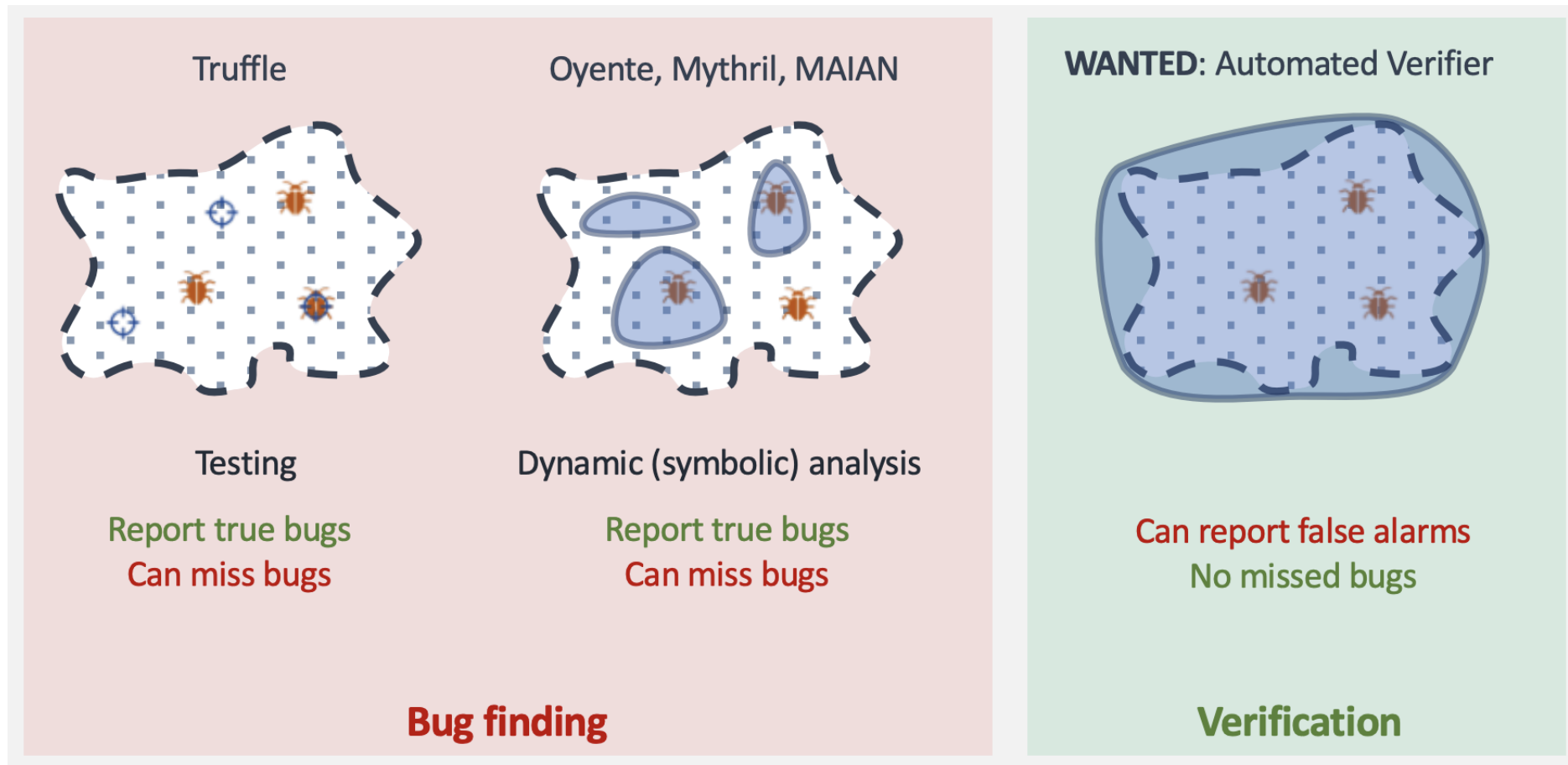


```
assert(c == a + b);
```

测试 VS 静态分析/程序验证



测试 VS 静态分析/程序验证




静态与动态分析方法的对比


- 静态方法不执行代码，动态方法会执行代码
- 静态分析
 - 准确率比动态方法低，必然存在false positive或false negative
- 形式化验证
 - 适用范围小，求解速度慢
- 测试
 - 只能找出小部分错误

程序验证

- 传统方法：形式化方法（静态）
 - 将程序抽象成一个数学公式，用严密的数学推理来证明（证实或证伪）该公式
- 一种形式化方法：模型检查
 - 判定代码是否满足规约（Specification）或不变式（Invariant）



```
1  contract InvariantDemo {  
2      uint c;  
3      function add(uint a, uint b) {  
4          c = a + b;  
5      }  
6  }
```




```
assert(c == a + b);
```



程序验证

能否用动态方法来做验证？

- 传统方法：形式化方法（静态）
 - 将程序抽象成一个数学公式，用严密的数学推理来证明（证实或证伪）该公式
- 一种形式化方法：模型检查
 - 判定代码是否满足规约（Specification）或不变式（Invariant）



```
1  contract InvariantDemo {  
2      uint c;  
3      function add(uint a, uint b) {  
4          c = a + b;  
5      }  
6  }
```



```
assert(c == a + b);
```

程序验证

能否用动态方法来做验证？

- 传统方法：形式化方法（静态）
 - 将程序抽象成一个数学公式，用严密的数学推理来证明（证实或证伪）该公式
- 一种形式化方法：模型检查
 - 判定代码是否满足规约（Specification）或不变式（Invariant）

```
1  contract InvariantDemo {  
2      uint c;  
3      function add(uint a, uint b) {  
4          c = a + b;  
5          assert(c == a + b);  
6      }  
7  }
```

智能合约动态验证

Securing Smart Contract with Runtime Validation

Ao Li, Jemin Andrew Choi, Fan Long - University of Toronto

王子彦

ziyan-wang.github.io

2021/06/24

Introduction

- 作者设计了一种源代码到源代码的Solidity编译器——Solythesis，用来对智能合约进行运行时的高效动态验证，确保合约的安全性
- 编译器接受Solidity合约源码和**不变式**，输出修改后的合约，该合约将拒绝所有违反不变式的交易

未经验证的普通合约（Solidity 语言）

```
1  contract InvariantDemo {
2      uint c;
3      function add(uint a, uint b) {
4          c = a + b;
5      }
6  }
```

编译器

修改后的安全合约（Solidity 语言）

```
1  contract InvariantDemo {
2      uint c;
3      function add(uint a, uint b) {
4          c = a + b;
5          assert(c == a + b);
6      }
7  }
```

`assert(c == a + b);`

不变式，即正确执行的判断条件
(论文作者设计的规范语言)

用Solidity官方编译器
编译成字节码

部署上链

运行时验证的性能问题？

- 运行时验证的开销对其他领域来说非常昂贵，但对智能合约来说可以忽略
- 经实验发现，区块链的PoW共识机制是性能的最大瓶颈
 - 一个以太坊节点每秒可处理34个交易
 - 不执行PoW共识的情况下，每秒可处理2181个交易
- 存储是性能的第二大瓶颈
 - 合约执行引擎花费67%的时间进行磁盘读写操作

```
1  contract InvariantDemo {  
2      uint c;  
3      function add(uint a, uint b) {  
4          c = a + b;  
5      }  
6  }
```

接下来用一个例子说明论文的思路

```
1  contract VoteExample {
2
3      mapping(address /*voter*/ => uint /*weight*/) public weights;
4      mapping(address /*voter*/ => uint /*option*/) public ballots;
5      mapping(uint /*option*/ => uint /*count*/) public weightedVoteCounts;
6
7      function vote(uint option) public {
8
9
10         weightedVoteCounts[option] += weights[msg.sender];
11         ballots[msg.sender] = option;
12     }
13 }
```

```
1  contract VoteExample {
2
3      mapping(address /*voter*/ => uint /*weight*/) public weights;
4      mapping(address /*voter*/ => uint /*option*/) public ballots;
5      mapping(uint /*option*/ => uint /*count*/) public weightedVoteCounts;
6
7      function vote(uint option) public {
8
9
10         weightedVoteCounts[option] += weights[msg.sender];
11         ballots[msg.sender] = option;
12     }
13 }
```

权重

投票选项记录

票数记录

统计票数

记录该用户投了哪个选项


```

1  contract VoteExample {
2
3      mapping(address /*voter*/ => uint /*weight*/) public weights;
4      mapping(address /*voter*/ => uint /*option*/) public ballots;
5      mapping(uint /*option*/ => uint /*count*/) public weightedVoteCounts;
6
7      function vote(uint option) public {
8
9
10         weightedVoteCounts[option] += weights[msg.sender];
11         ballots[msg.sender] = option;
12     }
13 }

```

权重

投票选项记录

票数记录

统计票数

记录该用户投了哪个选项

Bug?

```

1  contract VoteExample {
2
3      mapping(address /*voter*/ => uint /*weight*/) public weights;
4      mapping(address /*voter*/ => uint /*option*/) public ballots;
5      mapping(uint /*option*/ => uint /*count*/) public weightedVoteCounts;
6
7      function vote(uint option) public {
8
9          weightedVoteCounts[ballots[msg.sender]] -= weights[msg.sender];
10         weightedVoteCounts[option] += weights[msg.sender];
11         assert(weightedVoteCounts[option] >= weights[msg.sender]);
12         ballots[msg.sender] = option;
13     }
14 }
15

```

权重

投票选项记录

票数记录

删除之前投过的票数

统计这次投的票数

防止溢出

记录该用户投了哪个选项

```
1  contract VoteExample {
2
3      mapping(address /*voter*/ => uint /*weight*/) public weights;
4      mapping(address /*voter*/ => uint /*option*/) public ballots;
5      mapping(uint /*option*/ => uint /*count*/) public weightedVoteCounts;
6
7      function vote(uint option) public {
8
9
10         weightedVoteCounts[option] += weights[msg.sender];
11         ballots[msg.sender] = option;
12     }
13 }
```

权重

投票选项记录

票数记录

统计票数

记录该用户投了哪个选项

Solythesis编译器的设计


- 智能合约的开发者需要人工编写该合约的不变式
- Solythesis编译器通过不变式指定的条件生成Solidity语句并插入代码，在执行时合约将检查是否满足不变式
- 增量更新、增量检查
 - 静态分析源代码，确定会被修改的变量和可能违反的不变式。然后插装代码来维护这些可能更改的值，仅检查可能违反的不变式

不变式语言

- 语句
 - 中间值声明（自由变量声明语句）
 - 约束声明（断言语句）
- 变量绑定

```
$votes = Map ($option) Sum weights[$voter] Over ($voter) Where ballots[$voter] == $option;  
ForAll ($option) Assert $option == 0 || $votes[$option] == weightedVoteCounts[$option];
```

```
mapping(address /*voter*/ => uint /*weight*/) public weights;  
mapping(address /*voter*/ => uint /*option*/) public ballots;  
mapping(uint /*option*/ => uint /*count*/) public weightedVoteCounts;
```




```
mapping(uint /*option*/ => uint /*count*/) $votes;
```

不变式语言

- 语句
 - 中间值声明（自由变量声明语句）
 - 约束声明（断言语句）
- 变量绑定

```
$votes = Map ($option) Sum weights[$voter] Over ($voter) Where ballots[$voter] == $option;  
ForAll ($option) Assert $option == 0 || $votes[$option] == weightedVoteCounts[$option];
```

```
mapping(address /*voter*/ => uint /*weight*/) public weights;  
mapping(address /*voter*/ => uint /*option*/) public ballots;  
mapping(uint /*option*/ => uint /*count*/) public weightedVoteCounts;
```



```
mapping(uint /*option*/ => uint /*count*/) $votes;
```

```

1  contract VoteExample {
2
3      mapping(address /*voter*/ => uint /*weight*/) public weights;
4      mapping(address /*voter*/ => uint /*option*/) public ballots;
5      mapping(uint /*option*/ => uint /*count*/) public weightedVoteCounts;
6
7      function vote(uint option) public {
8
9
10         weightedVoteCounts[option] += weights[msg.sender];
11         ballots[msg.sender] = option;
12     }
13 }

```

权重

投票选项记录

票数记录

统计票数

记录该用户投了哪个选项

不变式

```

$votes = Map ($option) Sum weights[$voter] Over ($voter) Where ballots[$voter] == $option;
ForAll ($option) Assert $option == 0 || $votes[$option] == weightedVoteCounts[$option];

```

```
$votes = Map ($option) Sum weights[$voter] Over ($voter) Where ballots[$voter] == $option;
ForAll ($option) Assert $option == 0 || $votes[$option] == weightedVoteCounts[$option];
```

中间值声明

约束声明

Input : Program P as a list of statements and a list of invariant rules R

Output: The instrumented program as a list of statements

```
1  $P' \leftarrow P$ 
2 for  $r \in R$  do
3   if  $r = "v = \text{Map } \vec{x} \text{ Sum } e \text{ Over } \vec{y} \text{ Where } c;"$ 
4     then
5       Assume  $v$  is fresh. Declare  $v$  in  $P'$ .
6       for  $s = "store\ a,\ _;" \in P$  do
7          $\mathcal{B} \leftarrow \text{BindExpr}(a, e) \cup \text{BindExpr}(a, c)$ 
8          $b \leftarrow \text{BindCond}(a, c)$ 
9          $pre \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
10           $t' = t - e; \text{store } v[\vec{x}], t'; \}"$ 
11          $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, b)$ 
12          $post \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
13           $t' = t + e; \text{store } v[\vec{x}], t'; \}"$ 
14          $post \leftarrow \text{Rewrite}(post, \mathcal{B}, b)$ 
15         Insert  $pre$  before  $s$  and  $post$  after  $s$  into  $P'$ 
16   else if  $r = "ForAll\ \vec{x}\ \text{Assert } c;"$  then
17     Declare a fresh map  $\alpha$  in  $P'$  corresponding to  $r$ 
18     for  $s = "store\ a,\ _;" \in P$  do
19        $\mathcal{B} \leftarrow \text{BindExpr}(a, c)$ 
20        $pre \leftarrow "\alpha[\vec{x}] = 1"$ 
21        $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, \langle \perp, \perp \rangle)$ 
22       Insert  $pre$  before  $s$  in  $P'$ 
23    $P \leftarrow P'$ 
24 for  $r = "ForAll\ \vec{x}\ \text{Assert } c;" \in R$  do
25    $\alpha \leftarrow$  The defined map that corresponds to  $r$ 
26    $s' \leftarrow "for\ \vec{x}\ \text{in } \alpha\ \{ \text{assert } c; \}"$ 
27   Insert  $s'$  at the end of  $P'$ 
28 return  $P'$ 
```



```
$votes = Map ($option) Sum weights[$voter] Over ($voter) Where ballots[$voter] == $option;
ForAll ($option) Assert $option == 0 || $votes[$option] == weightedVoteCounts[$option];
```

中间值声明

约束声明

Input : Program P as a list of statements and a list of invariant rules R

Output : The instrumented program as a list of statements

```
1  $P' \leftarrow P$ 
2 for  $r \in R$  do
3   if  $r = "v = \text{Map } \vec{x} \text{ Sum } e \text{ Over } \vec{y} \text{ Where } c;"$   $\leftarrow$  处理中间值声明
4     then
5       Assume  $v$  is fresh. Declare  $v$  in  $P'$ .
6       for  $s = "store\ a,\ _;" \in P$  do
7          $\mathcal{B} \leftarrow \text{BindExpr}(a, e) \cup \text{BindExpr}(a, c)$ 
8          $b \leftarrow \text{BindCond}(a, c)$ 
9          $pre \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
10         $t' = t - e; store\ v[\vec{x}], t'; \}"$ 
11         $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, b)$ 
12         $post \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
13         $t' = t + e; store\ v[\vec{x}], t'; \}"$ 
14         $post \leftarrow \text{Rewrite}(post, \mathcal{B}, b)$ 
15        Insert  $pre$  before  $s$  and  $post$  after  $s$  into  $P'$   $\leftarrow$  在语句上方和下方插入相应的维护代码
16   else if  $r = "ForAll\ \vec{x}\ Assert\ c;"$  then  $\leftarrow$  处理约束声明
17     Declare a fresh map  $\alpha$  in  $P'$  corresponding to  $r$ 
18     for  $s = "store\ a,\ _;" \in P$  do
19        $\mathcal{B} \leftarrow \text{BindExpr}(a, c)$ 
20        $pre \leftarrow "\alpha[\vec{x}] = 1"$ 
21        $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, \langle \perp, \perp \rangle)$ 
22       Insert  $pre$  before  $s$  in  $P'$   $\leftarrow$  在语句上方插入相应的维护代码
23    $P \leftarrow P'$ 
24   for  $r = "ForAll\ \vec{x}\ Assert\ c;" \in R$  do
25      $\alpha \leftarrow$  The defined map that corresponds to  $r$ 
26      $s' \leftarrow "for\ \vec{x}\ in\ \alpha\ \{ assert\ c;\ }"$ 
27     Insert  $s'$  at the end of  $P'$   $\leftarrow$  在函数最后插入一个for循环, 验证合约状态是否符合约束声明
28 return  $P'$ 
```

```

$votes = Map ($option) Sum weights[$voter] Over ($voter) Where ballots[$voter] == $option;
ForAll ($option) Assert $option == 0 || $votes[$option] == weightedVoteCounts[$option];

```

中间值声明

约束声明

Input : Program P as a list of statements and a list of invariant rules R

Output : The instrumented program as a list of statements

```

1  $P' \leftarrow P$ 
2 for  $r \in R$  do
3   if  $r = "v = \text{Map } \vec{x} \text{ Sum } e \text{ Over } \vec{y} \text{ Where } c;"$ 
4     then
5       Assume  $v$  is fresh. Declare  $v$  in  $P'$ .
6       for  $s = "store\ a,\ _;" \in P$  do
7          $\mathcal{B} \leftarrow \text{BindExpr}(a, e) \cup \text{BindExpr}(a, c)$ 
8          $b \leftarrow \text{BindCond}(a, c)$ 
9          $pre \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
10           $t' = t - e; store\ v[\vec{x}], t'; \}"$ 
11          $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, b)$ 
12          $post \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
13           $t' = t + e; store\ v[\vec{x}], t'; \}"$ 
14          $post \leftarrow \text{Rewrite}(post, \mathcal{B}, b)$ 
15         Insert  $pre$  before  $s$  and  $post$  after  $s$  into  $P'$ 
16       else if  $r = "ForAll\ \vec{x}\ Assert\ c;"$  then
17         Declare a fresh map  $\alpha$  in  $P'$  corresponding to  $r$ 
18         for  $s = "store\ a,\ _;" \in P$  do
19            $\mathcal{B} \leftarrow \text{BindExpr}(a, c)$ 
20            $pre \leftarrow "\alpha[\vec{x}] = 1"$ 
21            $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, \langle \perp, \perp \rangle)$ 
22           Insert  $pre$  before  $s$  in  $P'$ 
23        $P \leftarrow P'$ 
24   for  $r = "ForAll\ \vec{x}\ Assert\ c;" \in R$  do
25      $\alpha \leftarrow$  The defined map that corresponds to  $r$ 
26      $s' \leftarrow "for\ \vec{x}\ in\ \alpha\ \{ assert\ c;\ }"$ 
27     Insert  $s'$  at the end of  $P'$ 
28 return  $P'$ 

```

把中间值声明中的变量映射（绑定）到代码中的变量

```
$votes = Map ($option) Sum weights[$voter] Over ($voter) Where ballots[$voter] == $option;
ForAll ($option) Assert $option == 0 || $votes[$option] == weightedVoteCounts[$option];
```

中间值声明

约束声明

Input : Program P as a list of statements and a list of invariant rules R

Output : The instrumented program as a list of statements

```
1  $P' \leftarrow P$ 
2 for  $r \in R$  do
3   if  $r = "v = \text{Map } \vec{x} \text{ Sum } e \text{ Over } \vec{y} \text{ Where } c;"$ 
4     then
5       Assume  $v$  is fresh. Declare  $v$  in  $P'$ .
6       for  $s = "store a, _;" \in P$  do
7          $\mathcal{B} \leftarrow \text{BindExpr}(a, e) \cup \text{BindExpr}(a, c)$ 
8          $b \leftarrow \text{BindCond}(a, c)$ 
9          $pre \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
10           $t' = t - e; store\ v[\vec{x}], t'; \}"$ 
11          $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, b)$ 
12          $post \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
13           $t' = t + e; store\ v[\vec{x}], t'; \}"$ 
14          $post \leftarrow \text{Rewrite}(post, \mathcal{B}, b)$ 
15         Insert  $pre$  before  $s$  and  $post$  after  $s$  into  $P'$ 
16       else if  $r = "ForAll\ \vec{x}\ \text{Assert } c;"$  then
17         Declare a fresh map  $\alpha$  in  $P'$  corresponding to  $r$ 
18         for  $s = "store a, _;" \in P$  do
19            $\mathcal{B} \leftarrow \text{BindExpr}(a, c)$ 
20            $pre \leftarrow "\alpha[\vec{x}] = 1"$ 
21            $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, \langle \perp, \perp \rangle)$ 
22           Insert  $pre$  before  $s$  in  $P'$ 
23        $P \leftarrow P'$ 
24   for  $r = "ForAll\ \vec{x}\ \text{Assert } c;" \in R$  do
25      $\alpha \leftarrow$  The defined map that corresponds to  $r$ 
26      $s' \leftarrow "for\ \vec{x}\ \text{in } \alpha\ \{ \text{assert } c; \}"$ 
27     Insert  $s'$  at the end of  $P'$ 
28 return  $P'$ 
```

合约中的语句（部分）

ballots[msg.sender]

不变式中的语句（部分）

ballots[\$voter] == \$option

把中间值声明中的变量映射（绑定）到代码中的变量

```
$votes = Map ($option) Sum weights[$voter] Over ($voter) Where ballots[$voter] == $option;
ForAll ($option) Assert $option == 0 || $votes[$option] == weightedVoteCounts[$option];
```

中间值声明

约束声明

Input : Program P as a list of statements and a list of invariant rules R

Output : The instrumented program as a list of statements

```
1  $P' \leftarrow P$ 
2 for  $r \in R$  do
3   if  $r = "v = \text{Map } \vec{x} \text{ Sum } e \text{ Over } \vec{y} \text{ Where } c;"$ 
4     then
5       Assume  $v$  is fresh. Declare  $v$  in  $P'$ .
6       for  $s = "store\ a,\ _;" \in P$  do
7          $\mathcal{B} \leftarrow \text{BindExpr}(a, e) \cup \text{BindExpr}(a, c)$ 
8          $b \leftarrow \text{BindCond}(a, c)$ 
9          $pre \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
10           $t' = t - e; store\ v[\vec{x}], t'; \}"$ 
11          $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, b)$ 
12          $post \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
13           $t' = t + e; store\ v[\vec{x}], t'; \}"$ 
14          $post \leftarrow \text{Rewrite}(post, \mathcal{B}, b)$ 
15         Insert  $pre$  before  $s$  and  $post$  after  $s$  into  $P'$ 
16       else if  $r = "ForAll\ \vec{x}\ Assert\ c;"$  then
17         Declare a fresh map  $\alpha$  in  $P'$  corresponding to  $r$ 
18         for  $s = "store\ a,\ _;" \in P$  do
19            $\mathcal{B} \leftarrow \text{BindExpr}(a, c)$ 
20            $pre \leftarrow "\alpha[\vec{x}] = 1"$ 
21            $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, \langle \perp, \perp \rangle)$ 
22           Insert  $pre$  before  $s$  in  $P'$ 
23        $P \leftarrow P'$ 
24   for  $r = "ForAll\ \vec{x}\ Assert\ c;" \in R$  do
25      $\alpha \leftarrow$  The defined map that corresponds to  $r$ 
26      $s' \leftarrow "for\ \vec{x}\ in\ \alpha\ \{ assert\ c;\ }"$ 
27     Insert  $s'$  at the end of  $P'$ 
28 return  $P'$ 
```

合约中的语句 (部分)

ballots[msg.sender]

不变式中的语句 (部分)

ballots[\$voter] == \$option

把中间值声明中的变量映射 (绑定) 到代码中的变量

$\text{BindExpr}(\text{ballots}[\text{msg.sender}], \text{ballots}[\$voter] == \$option)$

$= \{ \$voter \leftrightarrow \text{msg.sender},$
 $\$option \leftrightarrow \text{ballots}[\text{msg.sender}] \}$

```
$votes = Map ($option) Sum weights[$voter] Over ($voter) Where ballots[$voter] == $option;
ForAll ($option) Assert $option == 0 || $votes[$option] == weightedVoteCounts[$option];
```

中间值声明

约束声明

Input : Program P as a list of statements and a list of invariant rules R

Output : The instrumented program as a list of statements

```
1  $P' \leftarrow P$ 
2 for  $r \in R$  do
3   if  $r = "v = \text{Map } \vec{x} \text{ Sum } e \text{ Over } \vec{y} \text{ Where } c;"$ 
4     then
5       Assume  $v$  is fresh. Declare  $v$  in  $P'$ .
6       for  $s = "store\ a,\ _;" \in P$  do
7          $\mathcal{B} \leftarrow \text{BindExpr}(a, e) \cup \text{BindExpr}(a, c)$ 
8          $b \leftarrow \text{BindCond}(a, c)$ 
9          $pre \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
10           $t' = t - e; store\ v[\vec{x}], t'; \}"$ 
11          $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, b)$ 
12          $post \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
13           $t' = t + e; store\ v[\vec{x}], t'; \}"$ 
14          $post \leftarrow \text{Rewrite}(post, \mathcal{B}, b)$ 
15         Insert  $pre$  before  $s$  and  $post$  after  $s$  into  $P'$ 
16       else if  $r = "ForAll\ \vec{x}\ Assert\ c;"$  then
17         Declare a fresh map  $\alpha$  in  $P'$  corresponding to  $r$ 
18         for  $s = "store\ a,\ _;" \in P$  do
19            $\mathcal{B} \leftarrow \text{BindExpr}(a, c)$ 
20            $pre \leftarrow "\alpha[\vec{x}] = 1"$ 
21            $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, \langle \perp, \perp \rangle)$ 
22           Insert  $pre$  before  $s$  in  $P'$ 
23        $P \leftarrow P'$ 
24   for  $r = "ForAll\ \vec{x}\ Assert\ c;" \in R$  do
25      $\alpha \leftarrow$  The defined map that corresponds to  $r$ 
26      $s' \leftarrow "for\ \vec{x}\ in\ \alpha\ \{ assert\ c;\ }"$ 
27     Insert  $s'$  at the end of  $P'$ 
28 return  $P'$ 
```

合约中的语句 (部分)

ballots[msg.sender]

不变式中的语句 (部分)

ballots[\$voter] == \$option

把中间值声明中的变量映射 (绑定) 到代码中的变量

BindExpr(ballots[msg.sender], ballots[\$voter] == \$option)

= { \$voter <-> msg.sender,
\$option <-> ballots[msg.sender] }

```
$votes = Map ($option) Sum weights[$voter] Over ($voter) Where ballots[$voter] == $option;
ForAll ($option) Assert $option == 0 || $votes[$option] == weightedVoteCounts[$option];
```

中间值声明

约束声明

Input : Program P as a list of statements and a list of invariant rules R

Output : The instrumented program as a list of statements

```
1  $P' \leftarrow P$ 
2 for  $r \in R$  do
3   if  $r = "v = \text{Map } \vec{x} \text{ Sum } e \text{ Over } \vec{y} \text{ Where } c;"$ 
4     then
5       Assume  $v$  is fresh. Declare  $v$  in  $P'$ .
6       for  $s = "store a, _;" \in P$  do
7          $\mathcal{B} \leftarrow \text{BindExpr}(a, e) \cup \text{BindExpr}(a, c)$ 
8          $b \leftarrow \text{BindCond}(a, c)$ 
9          $pre \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
10           $t' = t - e; store\ v[\vec{x}], t'; \}"$ 
11          $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, b)$ 
12          $post \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
13           $t' = t + e; store\ v[\vec{x}], t'; \}"$ 
14          $post \leftarrow \text{Rewrite}(post, \mathcal{B}, b)$ 
15         Insert  $pre$  before  $s$  and  $post$  after  $s$  into  $P'$ 
16       else if  $r = "ForAll\ \vec{x}\ Assert\ c;"$  then
17         Declare a fresh map  $\alpha$  in  $P'$  corresponding to  $r$ 
18         for  $s = "store a, _;" \in P$  do
19            $\mathcal{B} \leftarrow \text{BindExpr}(a, c)$ 
20            $pre \leftarrow "\alpha[\vec{x}] = 1"$ 
21            $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, \langle \perp, \perp \rangle)$ 
22           Insert  $pre$  before  $s$  in  $P'$ 
23        $P \leftarrow P'$ 
24   for  $r = "ForAll\ \vec{x}\ Assert\ c;" \in R$  do
25      $\alpha \leftarrow$  The defined map that corresponds to  $r$ 
26      $s' \leftarrow "for\ \vec{x}\ in\ \alpha\ \{ assert\ c;\ }"$ 
27     Insert  $s'$  at the end of  $P'$ 
28 return  $P'$ 
```

合约中的语句 (部分)

ballots[msg.sender]

不变式中的语句 (部分)

ballots[\$voter] == \$option

把中间值声明中的变量映射 (绑定) 到代码中的变量

BindExpr(ballots[msg.sender], ballots[\$voter] == \$option)

= { \$voter <-> msg.sender,
\$option <-> ballots[msg.sender] }

```
$votes = Map ($option) Sum weights[$voter] Over ($voter) Where ballots[$voter] == $option;
ForAll ($option) Assert $option == 0 || $votes[$option] == weightedVoteCounts[$option];
```

中间值声明

约束声明

Input : Program P as a list of statements and a list of invariant rules R

Output : The instrumented program as a list of statements

```
1  $P' \leftarrow P$ 
2 for  $r \in R$  do
3   if  $r = "v = \text{Map } \vec{x} \text{ Sum } e \text{ Over } \vec{y} \text{ Where } c;"$   $\leftarrow$  处理中间值声明
4     then
5       Assume  $v$  is fresh. Declare  $v$  in  $P'$ .
6       for  $s = "store\ a,\ _;" \in P$  do
7          $\mathcal{B} \leftarrow \text{BindExpr}(a, e) \cup \text{BindExpr}(a, c)$ 
8          $b \leftarrow \text{BindCond}(a, c)$ 
9          $pre \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
10         $t' = t - e; store\ v[\vec{x}], t'; \}"$ 
11         $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, b)$ 
12         $post \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
13         $t' = t + e; store\ v[\vec{x}], t'; \}"$ 
14         $post \leftarrow \text{Rewrite}(post, \mathcal{B}, b)$ 
15        Insert  $pre$  before  $s$  and  $post$  after  $s$  into  $P'$   $\leftarrow$  在语句上方和下方插入相应的维护代码
16   else if  $r = "ForAll\ \vec{x}\ Assert\ c;"$  then  $\leftarrow$  处理约束声明
17     Declare a fresh map  $\alpha$  in  $P'$  corresponding to  $r$ 
18     for  $s = "store\ a,\ _;" \in P$  do
19        $\mathcal{B} \leftarrow \text{BindExpr}(a, c)$ 
20        $pre \leftarrow "\alpha[\vec{x}] = 1"$ 
21        $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, \langle \perp, \perp \rangle)$ 
22       Insert  $pre$  before  $s$  in  $P'$ 
23    $P \leftarrow P'$ 
24   for  $r = "ForAll\ \vec{x}\ Assert\ c;" \in R$  do
25      $\alpha \leftarrow$  The defined map that corresponds to  $r$ 
26      $s' \leftarrow "for\ \vec{x}\ in\ \alpha\ \{ assert\ c;\ }"$ 
27     Insert  $s'$  at the end of  $P'$   $\leftarrow$  在函数最后插入一个for循环, 验证合约状态是否符合约束声明
28 return  $P'$ 
```



```

$votes = Map ($option) Sum weights[$voter] Over ($voter) Where ballots[$voter] == $option;
ForAll ($option) Assert $option == 0 || $votes[$option] == weightedVoteCounts[$option];

```

```

1  contract VoteExample {
2
3      mapping(address /*voter*/ => uint /*weight*/) public weights;
4      mapping(address /*voter*/ => uint /*option*/) public ballots;
5      mapping(uint /*option*/ => uint /*count*/) public weightedVoteCounts;
6
7      function vote(uint option) public {
8
9          weightedVoteCounts[option] += weights[msg.sender];
10         ballots[msg.sender] = option;
11     }
12 }
13

```

Input : Program P as a list of statements and a list of invariant rules R

Output: The instrumented program as a list of statements

```

1   $P' \leftarrow P$ 
2  for  $r \in R$  do
3      if  $r = "v = \text{Map } \vec{x} \text{ Sum } e \text{ Over } \vec{y} \text{ Where } c;"$ 
4          then
5              Assume  $v$  is fresh. Declare  $v$  in  $P'$ .
6              for  $s = "store\ a,\ _;" \in P$  do
7                   $\mathcal{B} \leftarrow \text{BindExpr}(a, e) \cup \text{BindExpr}(a, c)$ 
8                   $b \leftarrow \text{BindCond}(a, c)$ 
9                   $pre \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
10                      $t' = t - e; \text{store } v[\vec{x}], t'; \}"$ 
11                   $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, b)$ 
12                   $post \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
13                      $t' = t + e; \text{store } v[\vec{x}], t'; \}"$ 
14                   $post \leftarrow \text{Rewrite}(post, \mathcal{B}, b)$ 
15                  Insert  $pre$  before  $s$  and  $post$  after  $s$  into  $P'$ 
16
17          else if  $r = "ForAll\ \vec{x}\ Assert\ c;"$  then
18              Declare a fresh map  $\alpha$  in  $P'$  corresponding to  $r$ 
19              for  $s = "store\ a,\ _;" \in P$  do
20                   $\mathcal{B} \leftarrow \text{BindExpr}(a, c)$ 
21                   $pre \leftarrow "\alpha[\vec{x}] = 1"$ 
22                   $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, \langle \perp, \perp \rangle)$ 
23                  Insert  $pre$  before  $s$  in  $P'$ 
24
25           $P \leftarrow P'$ 
26
27  for  $r = "ForAll\ \vec{x}\ Assert\ c;" \in R$  do
28       $\alpha \leftarrow$  The defined map that corresponds to  $r$ 
29       $s' \leftarrow "for\ \vec{x}\ in\ \alpha\ \{ \text{assert } c; \}"$ 
30      Insert  $s'$  at the end of  $P'$ 
31
32  return  $P'$ 

```

56
Figure 8. Instrumentation algorithm.


```

$votes = Map ($option) Sum weights[$voter] Over ($voter) Where ballots[$voter] == $option;
ForAll ($option) Assert $option == 0 || $votes[$option] == weightedVoteCounts[$option];

```

```

1  contract VoteExample {
2
3      mapping(address /*voter*/ => uint /*weight*/) public weights;
4      mapping(address /*voter*/ => uint /*option*/) public ballots;
5      mapping(uint /*option*/ => uint /*count*/) public weightedVoteCounts;
6      mapping(uint /*option*/ => uint /*count*/) public $votes;
7      uint[] $optionChecks;
8
9      function vote(uint option) public {
10
11
12          $optionChecks.push(option);
13          weightedVoteCounts[option] += weights[msg.sender];
14
15          assert($votes[ballots[msg.sender]] >= weights[msg.sender]);
16          $optionChecks.push(ballots[msg.sender]);
17          $votes[ballots[msg.sender]] -= weights[msg.sender];
18
19          ballots[msg.sender] = option;
20          $optionChecks.push(ballots[msg.sender]);
21          $votes[ballots[msg.sender]] += weights[msg.sender];
22          assert($votes[ballots[msg.sender]] >= weights[msg.sender]);
23
24          for (uint256 $index = 0; $index < $optionChecks.length; $index += 1) {
25              uint $optionCheck = $optionChecks[$index];
26              assert($optionCheck == 0 || $votes[$optionCheck] == weightedVoteCounts[$optionCheck]);
27          }
28          delete $optionChecks;
29      }
30 }

```

Input : Program P as a list of statements and a list of invariant rules R

Output: The instrumented program as a list of statements

```

1   $P' \leftarrow P$ 
2  for  $r \in R$  do
3      if  $r = "v = \text{Map } \vec{x} \text{ Sum } e \text{ Over } \vec{y} \text{ Where } c;"$ 
4          then
5              Assume  $v$  is fresh. Declare  $v$  in  $P'$ .
6              for  $s = "store a, _;" \in P$  do
7                   $\mathcal{B} \leftarrow \text{BindExpr}(a, e) \cup \text{BindExpr}(a, c)$ 
8                   $b \leftarrow \text{BindCond}(a, c)$ 
9                   $pre \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
10                      $t' = t - e; \text{store } v[\vec{x}], t'; \}"$ 
11                   $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, b)$ 
12                   $post \leftarrow "if\ c\ \{ t = \text{load } v[\vec{x}];$ 
13                      $t' = t + e; \text{store } v[\vec{x}], t'; \}"$ 
14                   $post \leftarrow \text{Rewrite}(post, \mathcal{B}, b)$ 
15                  Insert  $pre$  before  $s$  and  $post$  after  $s$  into  $P'$ 
16
17      else if  $r = "ForAll\ \vec{x}\ \text{Assert } c;"$  then
18          Declare a fresh map  $\alpha$  in  $P'$  corresponding to  $r$ 
19          for  $s = "store a, _;" \in P$  do
20               $\mathcal{B} \leftarrow \text{BindExpr}(a, c)$ 
21               $pre \leftarrow "\alpha[\vec{x}] = 1"$ 
22               $pre \leftarrow \text{Rewrite}(pre, \mathcal{B}, \langle \perp, \perp \rangle)$ 
23              Insert  $pre$  before  $s$  in  $P'$ 
24
25       $P \leftarrow P'$ 
26
27  for  $r = "ForAll\ \vec{x}\ \text{Assert } c;" \in R$  do
28       $\alpha \leftarrow$  The defined map that corresponds to  $r$ 
29       $s' \leftarrow "for\ \vec{x}\ \text{in } \alpha\ \{ \text{assert } c; \}"$ 
30      Insert  $s'$  at the end of  $P'$ 
31
32  return  $P'$ 

```

57
Figure 8. Instrumentation algorithm.

Future Directions

- 可以为智能合约设计内置动态验证功能的新语言

Future Directions

- 可以为智能合约设计内置动态验证功能的新语言

```
/* Concatenate two lists 'a' and 'b'.  
 * The result is a single list 'res'. */  
procedure concat(a: Node, b: Node)  
  returns (res: Node)  
  requires list(a) &*& list(b)  
  ensures list(res)  
{  
  if (a == null) {  
    return b;  
  } else {  
    var curr := a;  
  
    while (curr.next != null)  
      invariant acc(curr) **~ list(a)  
      {  
        curr := curr.next;  
      }  
    curr.next := b;  
    return a;  
  }  
}
```

Future Directions

- 可以为智能合约设计内置动态验证功能的新语言
- 由于动态验证开销较小， 可以结合静态分析和动态验证来提高程序的安全性
- 可以在区块链的虚拟机内部进行动态验证， 提高性能， 但需要对现有的链进行硬分叉

Thanks

智能合约动态验证

Securing Smart Contract with Runtime Validation

Ao Li, Jemin Andrew Choi, Fan Long - University of Toronto

王子彦

ziyan-wang.github.io

2021/06/24