文章编号:1005-0523(2021)03-0052-09

基于模型检测的区块链智能合约公平性形式化验证

肖美华[1],周浩洋[1],朱志亮[1],罗 敏[2]

(1. 华东交通大学软件学院,江西 南昌 330013;2. 江西省计算技术研究所,江西 南昌 330003)

摘要:随着第二代区块链平台及应用的爆发式增长,作为部署在区块链上可执行代码的智能合约面临越来越多的安全性问题。但是目前针对智能合约安全性问题的研究大多集中在对安全漏洞的挖掘,很少关注智能合约本身公平性对安全性的影响,对此提出一种基于模型检测的智能合约公平性验证方法。采用该方法对Puzzle合约的公平性进行验证,找到了一个已知的交易顺序依赖漏洞。结果表明提出的方法可以为发现智能合约中存在的漏洞提供新的思路。

关键词:智能合约;形式化方法;模型检测;交易顺序依赖漏洞

中图分类号:TP311 **文献标志码**:A

本文引用格式:肖美华,周浩洋,朱志亮,等. 基于模型检测的区块链智能合约公平性形式化验证[J]. 华东交通大学学报,2021,38(3):52-60.

DOI:10.16749/j.cnki.jecjtu.20210706.009

Formal Verification of Fairness of Block Chain Smart Contract Based on Model Checking

Xiao Meihua[1],Zhou Haoyang[1],Zhu Zhiliang[1],Luo Min[2]

(1. School of Software, East China Jiaotong University,Nanchang 330013,China;
2. Jiangxi Institute of Computing Technology,Nanchang 330003,China)

Abstract: With the explosive growth of the second generation blockchain platforms and applications, smart contracts as executable code deployed in the blockchain are facing more and more security problems. At present, most of the researches on the security of smart contracts focus on the mining of security vulnerabilities, while insufficient attention has been paid to the impact of the fairness of smart contracts on security. To address this problem, a fairness verification method based on model checking is proposed, which is used to verify the fairness of Puzzle contract, and a known transaction order dependency vulnerability is found. The results show that the proposed method can provide a new idea for verifying the fairness of smart contracts.

Key words: smart contract; formal method; model checking; transaction order dependence vulnerability

Citation format: XIAO M H,ZHOU H Y,ZHU Z L,et al. Formal verification of fairness of block chain smart contract based on model checking[J]. Journal of East China Jiaotong University,2021,38(3):52-60.

自2009年中本聪[1]引入比特币以来,分布式加密货币已经获得国内外学者的关注。加密货币由用户在其网络中公开管理,且不依赖于任何可信方,用户采用共识协议来维护共享的数据分类帐(区块

收稿日期:2021-03-06

基金项目:国家自然科学基金(61962020,61562026);江西省主要学科学术和技术带头人资助计划(20172BCB22015);江西省研究生创新专项基金(YC2019-S251);江西省青年科学基金资助项目(20202BAAL212006)

作者简介:肖美华(1967—),男,教授,博士,博士生导师,研究方向为形式化方法、区块链。E-mail:xiaomh@ecjtu.edu.cn。

通信作者:周浩洋(1994—),男,硕士研究生,研究方向为网络安全、形式化方法、区块链。E-mail:hy_777666@163.com。

链)。区块链[2]技术因为其去中心化的特性为人们所熟知,同时,基于区块链技术更广泛的应用被人们开发出来[3],智能合约就是其中之一。

智能合约可以对其编程语言中表示的任何一组规则进行编码[4-5],其与传统行业的结合可以提高该行业的效率以及安全性。肖谦等[6]研究中国分布式电力交易的实际情况,提出了一种基于区块链智能合约的分散式电力交易市场框架。徐美强等[7]基于区块链技术设计了变电站数字化配置的自动化方案,提高了效率和安全性。刘维扬等[8]基于智能合约技术提出一种电动汽车入网竞价机制,在实现电网负荷削峰填谷的同时,有效保障了用户、代理商和电力调度中心的利益。

由于智能合约应用便利,大量高价值数字资产利用智能合约进行存储和转移,容易受到攻击者的密集活动影响。随着智能合约安全受到越来越多的重视,国内外公司和研究机构也在致力于寻找智能合约安全性验证的方法。Bhargavan 等[9]将 Solidity 代码和 EVM 字节码转换成 F*, 然后使用 F* 类型检查来验证智能合约内存在的重大漏洞,但是其并不能完整的转换 Solidity 代码。Nehai 等[10]将智能合约翻译成 NuSMV 输入语言, 然后使用模型检测器对智能合约进行验证,但是其转换规则并不适合复杂的智能合约。Bigi 等[11]将博弈论用在智能合约形式化验证中,使用概率模型检测对智能合约进行验证, 但是只能针对有不确定性人类行为的智能合约。Luu 等[12]使用 Oyente 工具可检测部分重要类型的漏洞,但不能涵盖所有已知类型的漏洞,检测出来的结果需要人工进行二次审计和确认。

智能合约作为部署在区块链系统的合同,其公平性是一个至关重要的属性,只有保证参与合约的各方都能得到自己应有的利益,才能使智能合约可信,从而使得智能合约可以被广泛应用。智能合约的公平性问题大多是由于合约内部的逻辑造成的,而每一个合约的内部逻辑都不相同,这使得此类问题的检测特别具有挑战性。

本文将智能合约间互相调用的过程抽象为主体间互相发送消息的协议,用进程表示主体,利用进程间通信模拟智能合约的互相调用过程。通过进程的并发执行来实现区块链环境中智能合约的并发调用过程,然后使用模型检测[13]方法对进程执行过程中的状态进行检测。从而发现智能合约调用过程中存在违反公平性约束的状态, 进而根据模型检测器产生的反例来推出智能合约中存在的漏洞。

1　智能合约公平性验证方法

1.1　方法概述

智能合约是根据文本合同编写的可以部署的区块链上的可执行代码。智能合约从生成到被调用一共包括以下几个阶段。

合约双方或多方达成共识->文本合同->智能合约代码->智能合约字节码->调用执行结果。

需要解决的是,由于智能合约代码和文本合同之间的不一致导致其执行结果与合同参与方预期不一致,从而使其中的某一方或多方利益受损的问题。对智能合约进行形式化验证时,需要满足 3 个条件。

1) 根据智能合约代码所建立的形式化模型必须准确的描述合约代码的内在逻辑。

2) 在刻画合约公平性时,需要根据合约文本来抽象出满足合约公平性的条件,不能直接从智能合约代码中对公平性进行抽象,因为可能存在智能合约代码本身的逻辑是与文本合约不一致的情况。

3) 在对形式化模型进行验证时,需要考虑真实合约执行过程中所有可能出现的情况,例如变量值可能极大,极小,或者出现负值,合约的调用可能会并发执行等等。

针对上述问题,提出基于模型检测的智能合约公平性验证方法,将智能合约间的互相调用抽象成为主体互相传递消息的协议,将合约内在逻辑抽象成为一个状态迁移函数,同时对智能合约应该满足的公平性用线性时态逻辑[14](linear temporal logic,LTL)公式进行形式化描述,通过对协议执行过程中各个主体的状态进行检测,发现违反智能合约公平性约束的状态以及对应的漏洞。具体验证过程如图 1 所示。

1.2　静态分析

在对智能合约进行建模之前,首先对智能合约代码进行静态分析,以排除直观的错误。可以对照如表 1 所示的目前已知的智能合约中常见的漏洞,如时间戳依赖、错误的异常处理、整数溢出等。这些都可以通过对代码静态分析发现。

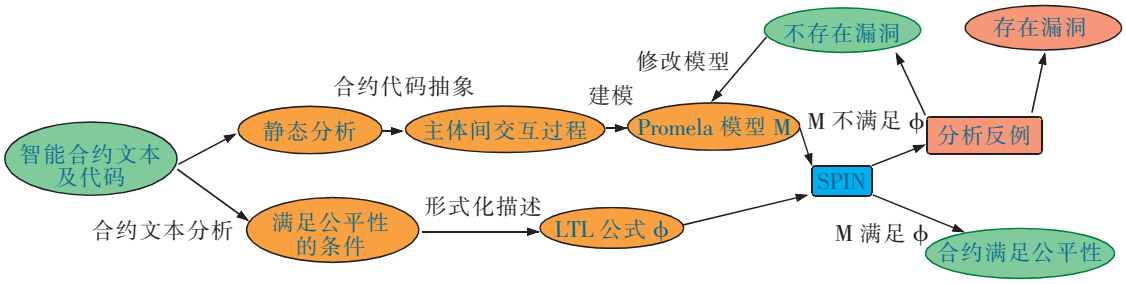

图 1 验证过程

Fig.1 Verification process

表 1 智能合约常见漏洞

Tab.1 Common vulnerabilities of smart contract

漏洞名称	漏洞机制	相关攻击
重入漏洞	从回退函数不停地递归调用函数	The DAO
交易顺序依赖	交易执行顺序与产生顺序不一致	–
时间戳依赖	使用块时间戳作为触发条件执行关键操作	–
错误的异常处理	未能在函数调用后检查返回值	The DAO,Integer Over/Under flow attack
调用堆栈深度限制	超过对约定方法调用次数的限制	–
整数溢出/下溢	从 0 减去正整数会得到大的值	Integer Over/Under flow attack
未检查且失败的发送	发送 Ether 时未检查条件	The DAO
不受限制的写	对存储变量的写入受到修饰符 private 的限制	Parity Multisig wallet attack
未验证参数	合约函数中的参数应该在使用之前进行验证	Integer Over/Under flow attack
Gas 超支	合约代码的执行不必要地消耗了更多的 Gas	–

1.3 Solidity 语义分析

为确保所建模型准确的描述智能合约代码的内在逻辑，需要对 Solidity 编程语言的语义进行分析[15]。由于通过静态分析已经剔除了那些如整数溢出、错误的异常处理等漏洞，所以建模时主要对合约的控制语句进行抽象，对 Solidity 语言的控制语句进行形式化定义，定义如下

$$\frac{(B,\sigma_B,\sigma_M)\rightarrow \mathrm{True}}{(\mathrm{if}(B)\{O\},\sigma_B,\sigma_M)\rightarrow(O,\sigma_B,\sigma_M)} \tag{1}$$

$$\frac{(B,\sigma_B,\sigma_M)\rightarrow \mathrm{False}}{(\mathrm{if}(B)\{O\},\sigma_B,\sigma_M)\rightarrow(\downarrow,\sigma_B,\sigma_M)} \tag{2}$$

$$\frac{(B,\sigma_B,\sigma_M)\rightarrow \mathrm{True}}{(\mathrm{while}(B)\{O\},\sigma_B,\sigma_M)\rightarrow(O,\mathrm{while}\{B\}\{O\},\sigma_B,\sigma_M)} \tag{3}$$

$$\frac{(B,\sigma_B,\sigma_M)\rightarrow \mathrm{False}}{(\mathrm{while}(B)\{O\},\sigma_B,\sigma_M)\rightarrow(\downarrow,\sigma_B,\sigma_M)} \tag{4}$$

对于 if 语句，B 表示语句的判断条件，σ_B,σ_M 则分别表示存储在区块链上的全局变量和存储在智能合约内部的局部变量，如果判断条件为真则执行操作 O，$\downarrow$ 表示判断条件为假时，不做任何操作。对于 while 语句，若判断条件为真，则执行循环体内部的操作语句然后再进行判断，若条件为假，则不做任何操作。在使用 Promela 进行建模时，需要将 Solidity 中的控制语句转换为 Promela 语句。定义转换规则如下

```
Solidity:(if(B),{O})|→
Promela:
if
    ::(B)→{O}
    ::else→skip
 fi
```
(5)

```
Solidity:(while(B),{O})|→
Promela:
 do
    ::(B)→{O}
    ::else→skip
 od
```
(6)

在对变量进行转换时,为 Solidity 中的每个全局变量定义一个对应的 Promela 变量,对于 Solidity 中的局部变量,则定义对应的主体名前缀的全局变量来标识。

1.4　合约调用过程抽象

由于公平性都是相对而言,所以在对智能合约调用过程进行抽象时,需要描述参与合约执行的全部主体。用主体间发送消息来模拟智能合约间互相发送交易的调用过程,消息内容包含交易信息以及交易所携带的金额。抽象后的主体间交互过程如图 2 所示。

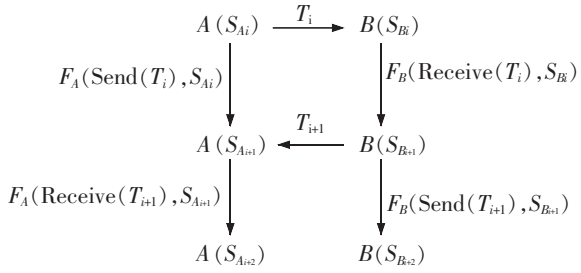

图2　主体间交互抽象

Fig.2　Interaction abstract between agents

以太坊中的智能合约可以看作是一个随着执行交易状态不断变换的有限状态机。所以在对智能合约的公平性进行验证时,把参与合约执行的主体看作一个整体。即智能合约的执行过程中是这个整体从一个状态到下一个状态的迁移过程。本文借鉴 Bai[16]中对智能合约的形式化定义并对其简化,将智能合约执行过程定义如式(7)所示的 4 元组

$$M=(S,T,A,F) \tag{7}$$

式中:S 为整体的当前状态,包括参与合约执行的全部主体状态,$S=(S_A,S_B,\cdots)$;T 为智能合约发送和接收的所有消息集合,$T=(T_1,T_2,\cdots)$;A 为智能合约主体所做的动作集合,包括 $\mathrm{Send}(T_i)$ 和 $\mathrm{Receive}(T_i)$,$A=(a_1,a_2,\ \cdots)$;F 为参与合约执行的主体内部函数的集合,$F=(F_A,F_B,\cdots)$。将主体的内部逻辑抽象为一个函数, 函数的输入为当前状态和所做的动作,输出为下一个状态,即 $S_{A_{i+1}}=F_A(S_{Ai},a_i)$。

虽然区块链上的所有计算都是确定性的,但是由于交易本身之间的竞争(即由矿工为给定的区块选择哪些交易),仍然会发生一定数量的不确定性。这里使用 Promela 语言中进程的并发执行来达到这个目的,使动作的执行顺序在一定的规则下存在随机性,以模拟智能合约真实的执行环境。

1.5　交互过程 Promela 建模

Promela 是一种描述并发系统的建模语言,用于有限状态机系统建模,是模型检测器 SPIN 的输入语言, 使用 Promela 进程间通信的方式对智能合约间互相调用的交互过程进行建模。在 Promela 中,主体间传递消息并不是直接发送消息给对方, 而是发送方将消息发送到消息通道内, 接收方从消息通道内取出消息。建立消息通道格式如式(8)所示

$$\text{chan ca=[0] of \{mtype,mtype\}} \tag{8}$$

式中:ca 为通道名称;[0] 为通道内可以存放的消息数量。如果是同步消息传递,则通道内可以存放的消息数量为 0, 异步传递则可以设置通道内存放的消息数量。{ }为消息内容,包括消息数据类型的定义以及数据项的数目。通道内发送消息和接收消息如下

$$\text{ca! } x1,x2 \tag{9}$$

$$\text{cb? } x1,x2 \tag{10}$$

式(9)表示向通道 ca 内发送一条消息,消息的内容包括 $x1$,$x2$ 两条数据。式(10)表示从通道 cb 内接收两条消息, 并将其分别赋值给变量 $x1$,$x2$。Promela 消息通道中还有一种用于判断的操作,如式(11)所示

$$\text{cb? eval}(x1),x2 \tag{11}$$

式(11)表示从通道 cb 内接收一条消息,如果第一项数据等于 $x1$,则把第二项数据赋值给变量 $x2$,否则丢弃这条消息。

1.6　智能合约公平性刻画

对于智能合约公平性的定义,不同的合约会有不同的描述。例如,一个简单的买卖合约,如果买方先付款,卖方后发货,就需要有对卖方不发货这种情况的惩罚性条款,否则对于买方是不公平的。在一个拍卖合约中,如果存在几个投标人私下串通然后再出价的情况,则会对其他投标人不公平。总的来说, 智能合约的公平性是指在合约执行过程中,可能出现的所有情况内,每一位合约参与者都能获得自己应有的利益。

在对具体合约安全性进行描述时, 需要结合文本合约以及所建的形式化模型来抽象出与公平性相关的变量。使用各个变量间的关系来表示公平性。在对公平性进行描述时, 要首先对智能合约执行流程进行分析,规定在一些特定的状态时,与公平性有关的单个或多个变量应该满足怎样的关系。然后使用 LTL 公式描述满足公平时变量之间的关系。

例如对于一个简单的转账操作中的变量之间的关系,公平性可以描述为转账前后,转出方和接收方两人的余额之和是恒定的。使用LTL描述如式(12)所示

[](balance[sender]+balance[receiver]) (12)

2 Puzzle合约公平性验证

Puzzle是一个在区块链上很常见的奖励合约。合约的主要功能是对向智能合约提交问题正确答案的用户发放奖励。这个合约的逻辑是首先智能合约向区块链网络中广播寻求问题解决答案的交易信息,交易内包含问题描述以及赏金的金额。然后用户节点向智能合约发送提交答案的交易。智能合约在收到答案后, 会对答案的正确性进行验证,若正确则向用户发放奖励,否则终止交易。同时智能合约的拥有者可以向智能合约发送修改奖励金额的交易,智能合约收到交易后会验证交易的发送方地址,如果是合约拥有者,则会将之前存放在智能合约的奖励金额返还给合约拥有者,并将拥有者发来的交易中携带的金额设置为新的奖励金额。经过静态分析修改后的Puzzle合约的核心代码如下:

```
contract Puzzle {
function (){ // 主函数,每次调用时执行
if (msg.sender==owner ){
owner.send (reward);
reward=msg.value; //修改奖励金额
}
else
if (msg.data.length > 0){ // 提交一个解决方案
if (sha256 (msg.data)< diff){//验证方案正确性
msg.sender.send (reward); //发放奖励
solution=msg.data;
}}}}
```

2.1 Puzzle合约抽象

参与Puzzle合约交互的主体有3个,分别是智能合约(contract)、合约拥有者(owner)和提交答案的用户(user)。在区块链网络中交易信息是公开的,所以可以简化加解密以及身份地址认证等操作,直接使用点对点通信的方式来表示交易信息的发送和接收,这样能够简化模型,避免模型检测过程中状态空间爆炸的问题。Puzzle合约主体间交互过程如图3所示。

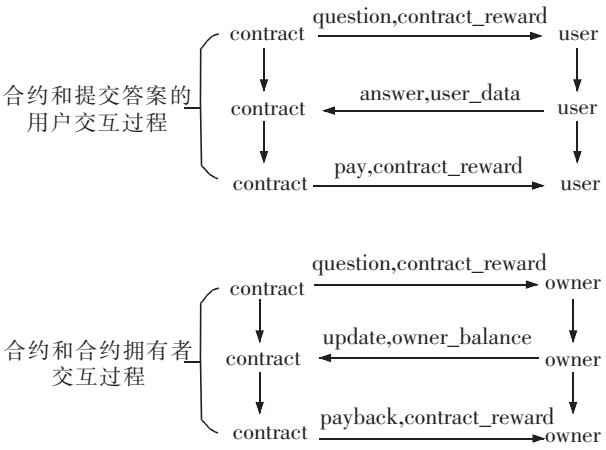

图3 主体间交互过程图

Fig.3 Interaction process diagram between agents

智能合约执行过程中的状态定义如式(13)所示

$$S=(Scontract,Sowner,Suser)=(contract\_reward,contract\_data,owner\_balance,owner\_reward,user\_balance,user\_reward,user\_data) \quad (13)$$

关于状态定义中每个变量的意义如表2所示。关于消息、函数及动作集合的定义将在下一节中使用Promela进程来说明。

表2 变量名称及意义

Tab.2 Name and meaning of each variable

变量名称	变量意义	初始值
contract_reward	智能合约内设置的奖励金额	5
contract_data	智能合约内存放的答案数据	0
owner_balance	合约拥有者的账户余额	4
owner_reward	合约拥有者收到的请求信息中表明的奖励金额	0
user_balance	用户账户余额	0
user_reward	用户收到的请求信息中表明的奖励金额	0
*user_data*0	表示正确的答案	1
*user_data*1	表示错误的答案	2

2.2 Puzzle 合约 Promela 建模

在对 Puzzle 合约交互过程抽象以后,可以发现有两对主体互相发送交易,分别是合约和用户以及合约和合约拥有者,通过定义两条一对一通信的消息通道来对进程间通信建模。为了便于解释下边的消息通道定义, 需要先说明有限名称集的定义,如式(14)所示。

mtype question,answer,update;pay,payback;contract_reward;user_data;owner_balance; (14)

前 5 个名称对应主体间发送消息的类型,分别是发送问题,提交答案,修改奖励金额,支付赏金以及退回奖励金额,后 3 个则对应消息中的变量。消息通道定义如式(15)所示。

chan ca1=[0] of {mtype,mtype};chan ca2=[0] of {mtype,mtype} (15)

通道 ca1 用于提交答案的用户和智能合约进行交互,通道 ca2 用于合约拥有者和智能合约进行交互。因为定义的通道是一对一的,所以在消息内部不需要再表明消息的发送者和接收者。

2.2.1 user 建模

使用进程 proctype user()来定义向智能合约提交答案的用户进程,在 proctype user()中主要对 *user_balance*,*user_reward*,*user_data*0,*user_data*1 这 4 个变量进行操作。user 进程主要做出的动作如式(16)所示。

$$\begin{aligned}&a1:receive(question,contract\_reward)from\ contract\\&a2:send(answer,user\_data0|\ user\_data1)to\ contract\\&a3:receive(pay,contract\_reward)from\ contract\end{aligned} \tag{16}$$

user 进程的内部逻辑则通过 Promela 语言中的控制语句以及消息通道内的判断操作来实现。这里将 user 进程内部的逻辑抽象为一个状态转移函数,函数的输入是 user 的当前状态以及所做的动作,输出为下一个状态。例如 user 的初始状态 $Suser_0$ 如式(17)所示。

$$Suser_0\ (user\_balance,user\_reward,user\_data0,user\_data1)=(0,0,1,2) \tag{17}$$

则经过执行动作 *a*1 后得状态 $Suser_0$ 为:$Suser_1=F_{user}(Suser_0,a1)=(0,5,1,2)$,鉴于篇幅原因,这里不再列出 user 进程的 Promela 代码。

2.2.2 contract 建模

使用进程 proctype contract()来定义智能合约进程,智能合约主体完成 3 个操作。

1) 向区块链网络广播消息,来请求问题答案。

2) 接收来自答案提交用户的交易信息,对答案验证后,按照最新的奖励金额向用户发放奖励。为了简化模型,省略了对答案验证的过程。

3) 接收来自合约拥有者的修改奖励金额的交易信息,将之前的奖励金返回给合约拥有者,然后将合约拥有者发来的交易中携带的金额设置为新的奖励金额。

使用两条发送语句模拟向网络中广播交易。智能合约主体的动作定义如式(18)所示。

$$\begin{aligned}&a1:send(question,contract\_reward)to\ user\\&a2:send(question,contract\_reward)to\ owner\\&a3:receive(answer,user\_data)from\ user\\&a4:receive(update,owner\_balance)from\ owner\\&a5:send(pay,contract\_reward)to\ user\\&a6:send(payback,contract\_reward)to\ owner\end{aligned} \tag{18}$$

智能合约主体内部函数的定义则使用 Promela 进程中关于通道内接收消息的判断操作以及控制语句来实现。智能合约主体内部逻辑核心代码如下所示。

```
    ca1!  question,contract_reward;
    ca2!  question,contract_reward;
do
  :: ca1 eval(answer), contract_data;
     if
       :: (contract_data==1)
            ->ca1!  pay,contract_reward;
       :: else -> skip;
     fi
  :: ca2 eval(update),a;
     ->ca2!  payback,contract_reward;
          contract_reward=a;
od
```

2.2.3 owner 建模

使用进程 proctype owner()来定义向智能合约提交答案的用户进程,在 proctype owner()中主要对 *owner_balance*,*owner_reward* 这 2 个变量进行操作。owner 进程主要做出的动作如式(19)所示。

$$\begin{aligned}&a1:receive(question,contract\_reward)from\ contract\\&a2:send(update,owner\_balance)to\ contract\\&a3:receive(payback,contract\_reward)from\ contract\end{aligned} \tag{19}$$

在这里假设合约拥有者在收到请求信息后,直

接向合约发送修改价格的交易信息。owner进程内部逻辑为,在收到请求信息后,将其中携带的奖励金额的数量减少1,作为新的奖励金额发送给合约。

2.3 Puzzle合约公平性刻画

在对智能合约的公平性进行刻画时,需要考虑智能合约是否完美的实现了它的功能且没有出现任何不允许出现的状态。在Puzzle合约中,要达到的目的就是提交答案的用户和合约拥有者能够公平的完成交易。因为省略了答案验证以及地址验证等操作,所以要公平的完成交易,就需要满足以下3点。

属性1 提交答案的用户在提交正确答案后,最终能够收到他接收问题时所观察到的奖励金额。即总是存在以下3种状态中的一种:

1) 用户还没有发送答案。

2) 用户发送了答案但是还没有收到钱。

3) 用户收到了他应该收到的奖励。

使用LTL公式描述如式(20)所示。

$$[]((\text{user\_data0}) \| (!\text{user\_balance}) \| (\text{fair\_user})) \quad (20)$$

fair_user用如下代码表示

```
if
  ::(user_reward==user_balance)
      -> fair_user=1;
  :: else->fair_user=0;
fi
```

属性2 合约拥有者在发送了新的奖励金额给智能合约后,安全的收到智能合约返还的奖励金额,且与他所观察到的修改前的奖励金额相等。使用LTL公式描述如式(21)所示。

$$<>\text{fair\_owner} \quad (21)$$

fair_owner用如下代码表示

```
if
  ::(owner_reward==owner_balance)
      -> fair_owner=1;
  :: else->fair_owner=0;
fi
```

属性3 合约在向用户发送完奖励以后,合约内一定存有正确的答案。这里使用*user_balance!*=0来表示合约向用户付款成功。即*contract_data*一定在*user_balance!*=0变为真之前为真。使用LTL公式描述如式(22)所示。

$$!\text{user\_balance}!=0 U \text{contract\_data}==1 \quad (22)$$

将上述3个属性的描述合成如式(23)所示。

$$\text{ltl t1}\{[]((\text{user\_data0}) \| (!\text{user\_balance}) \| (\text{fair\_user}) \&\& (<>\text{fair\_owner}) \&\& (!\text{user\_balance})=0 U \text{contract\_data}==1)\} \quad (23)$$

2.4 验证结果分析

将上述模型在SPIN6.4.9,Ispin 1.1.4中运行,验证结果表明在搜索深度为33时,公平性约束被违反,查看深度33时各个变量的值如表3所示。

表3 搜索深度为33时各个变量值

Tab.3 The value of each variable at depth 33

变量名称	变量值
contract_data	1
contract_reward	4
fair_owner	1
fair_user	0
owner_balance	5
user_balance	4
*user_data*0	0
*user_data*1	2
user_reward	5

在深度33时*owner_balance*与*owner_reward*相等,即拥有者的公平得到了保证,但是当*user_data*0=0,即用户提交了答案以后,用户最后得到的奖励与他接收问题时所观察到的不同,*user_balance*不等于*user_reward*,即用户的公平性没有得到保证。查看违反公平性时的消息序列如图4所示。当公平性被违反时,主体间交互过程如图5所示。

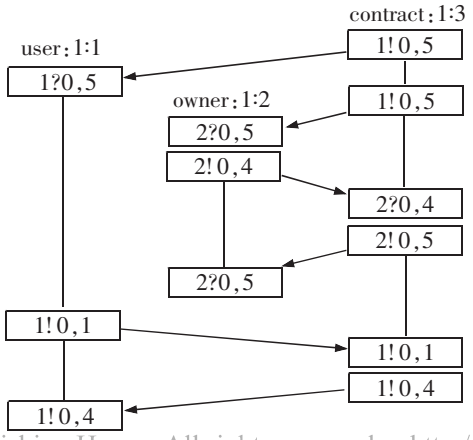

图4 公平性违反时消息序列

Fig.4 Message sequence graph for fairness violation

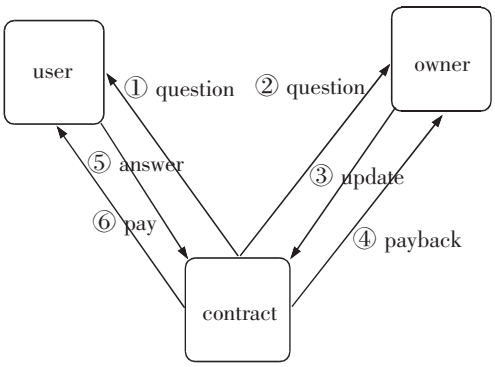

图 5　主体间交互过程
Fig.5　Interaction process between agents

为了更加细致的分析造成公平性被违反的原因,需要根据主体间的交互过程来刻画智能合约的状态迁移序列。Puzzle 合约状态迁移如图 6 所示(*标记值发生变化的变量),这里仅列出与公平性有关的变量。

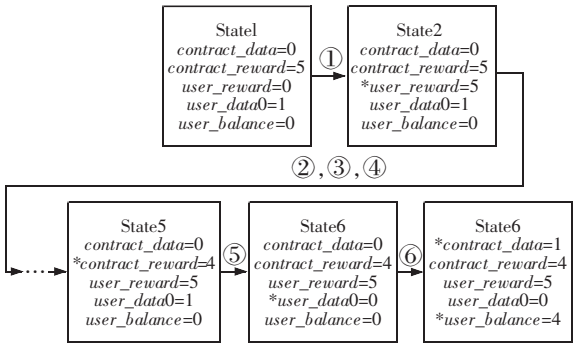

图 6　Puzzle 合约公平性违反时状态迁移图
Fig.6　State transition graph for fairness violation of Puzzle contract

由此可知 Puzzle 合约不满足公平性是因为用户和合约拥有者两个主体同时发送交易调用了智能合约,由于同一区块内交易执行顺序的随机性,导致用户接收的奖励是被合约拥有者修改以后的奖励金额。即用户没有收到他接收问题时所看到的奖励金额。结合文献[12]中对交易顺序依赖漏洞的分析,Puzzle 合约中存在交易顺序依赖漏洞,对提交答案的用户是不公平的。通过上述实验证明,发现 Puzzle 合约中存在公平性漏洞,证明了本文中提出方法的可行性。

3　结论

本文提出一种基于模型检测的智能合约公平性验证方法,将智能合约执行过程抽象为主体间交互的协议,模拟智能合约间并发调用的过程,使用 LTL 公式对智能合约需要满足的公平属性进行刻画,使用模型检测器 SPIN 对智能合约公平性进行验证。

1)解决了使用模型检测方法对智能合约进行验证时,所建立的模型只能针对一种类型合约的问题,为发现智能合约并发调用时产生的漏洞提供了新的方法。使用该方法对 Puzzle 合约公平性进行验证,发现该合约存在交易顺序依赖漏洞。

2)本方法目前仅适用于使用 Solidity 编写的智能合约,未来工作将致力于扩展语言转换规则,以及属性刻画方法。使其可以对其他语言编写的智能合约的更多属性进行验证以致于可以发现更多类型的漏洞,同时也将致力于实现对智能合约主体间交互过程建模的自动化。

参考文献:

[1] NAKAMOTO S. Bitcoin:A peer-to-peer electronic cash system[EB/OL].(2008-10-31)[2020-11-19].https://bitcoin-cash.org/bitcoin.pdf.

[2] ZHENG Z,XIE S,DAI H N,et al. Blockchain challenges and opportunities:A survey[J]. International Journal of Web and Grid Services,2018,14(4):352-375.

[3] 张利华,付东辉,万源华. 基于区块链的医疗记录安全共享方案[J]. 华东交通大学学报,2020,37(5):124-129.

[4] 夏清,窦文生,郭凯文,等.区块链共识协议综述[J]. 软件学报,2021,32(2):277-299.

[5] CHRISTIDIS K,DEVETSIKIOTIS M. Blockchains and smart contracts for the internet of things[J]. IEEE Access,2016,4:2292-2303.

[6] 肖谦,陈政,朱宗耀,等. 适应分布式发电交易的分散式电力市场探讨[J]. 电气系统自动化,2020,44(1):208-218.

[7] 徐美强,高志远,王伟,等. 基于区块链技术的智能变电站配置版本管理[J]. 电力系统保护与控制,2020,48(2):60-67.

[8] 刘维扬,王冰,王敏,等. 智能合约技术下电动汽车入网竞价机制研究[J]. 电网技术,2019,43(12):4344-4352.

[9] BHARGAVAN K,SWAMY N,SANTIAGO Z B,et al. Formal verification of smart contracts:Short paper[C]//Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security. New York:Association for Computing Machinery,2016.

[10] NEHAI Z,PIRIOU P Y,DAUMAS F. Model-checking of

smart contracts[C]//2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cybar,Physical and Social Computing and IEEE Smart Data. Halifax,Canada:IEEE,2018.

[11] BIGI G,BRACCIALI A,MEACCI G,et al. Validation of decentralised smart contracts through game theory and formal methods[J]. Springer International Publishing,2015:142-161.

[12] LUU L,CHU D H,OLICKEL H, et al. Making smart contracts smarter[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. Vienna,Austria:ACM,2016.

[13] 肖美华. 安全协议形式化分析与验证[M]. 北京:科学出版社,2020.

[14] 钟小妹,肖美华,李伟,等. RFID超轻量级认证协议RCIA形式化分析与改进 [J]. 计算机工程与科学,2018,40(12):2183-2192.

[15] OSTERLAND T,ROSE T. Model checking smart contracts for ethereum[J]. Pervasive and Mobile Computing,2020,63:101-129.

[16] BAI X M,CHENG Z J,DUAN Z B,et al. Formal modeling and verifcation of smart contracts[C]//Proceedings of the 2018 7th International Conference on Software and Computer Applications. Kuatan,Malaysia:ICSCA,2018.