张取发,王昌晶,左正康,等. 一种基于 UPPAAL 的智能合约属性形式化验证方法[J]. 江西师范大学学报(自然科学版),2023,47(1):45-51.
ZHANG Qufa,WANG Changjing,ZUO Zhengkang,et al. The formal verification method for smart contract properties based on UPPAAL[J]. Journal of Jiangxi Normal University(Natural Science),2023,47(1):45-51.

文章编号:1000-5862(2023)01-0045-07

一种基于 UPPAAL 的智能合约属性形式化验证方法

张取发[1],王昌晶[1,2*],左正康[1],卢家兴[1*],廖云燕[1],王 渊[3]
(1. 江西师范大学计算机信息工程学院,江西 南昌 330022;2. 江西师范大学管理科学与工程研究中心,江西 南昌 330022;
3. 江西师范大学软件学院,江西 南昌 330027)

摘要:针对智能合约的属性验证问题,该文提出了一种基于 UPPAAL 的智能合约属性形式化验证方法. 首先定义了 Solidity 基本语句的操作语义及其到时间自动机的转换,将智能合约转换成时间自动机网络模型;然后定义并描述智能合约常见的安全性和活性,再使用模型检测工具 UPPAAL 验证智能合约的属性;最后对购物合约进行了建模与验证,验证了该方法的有效性.

关键词:智能合约;UPPAAL;时间自动机;安全性;活性
中图分类号:TP 311 **文献标志码**:A **DOI**:10.16357/j.cnki.issn1000-5862.2023.01.06

0 引言

区块链是一种安全共享的去中心化的数据账本,智能合约是部署在区块链网络上的计算机程序[1],当满足必要条件时,它就会自动执行. 早在 1994 年,N. Szabo[2] 就提出了智能合约的概念,他指出“一个智能合约是一套数字形式定义的承诺,包括合约的参与方可以在上面执行这些承诺的协议”. 随着具有去中心化和公开透明等特性的区块链技术出现,智能合约的信任问题得到了有效解决. 目前,以太坊[3] 已成为主流的智能合约开发和运行的平台. 智能合约包含了区块链的众多特性,使得区块链与智能合约技术在许多领域都得到了广泛的应用.

尽管智能合约发展迅速,但智能合约的执行代码不能保证绝对安全,且智能合约运行时往往会伴随着大量数字资产的存储和转移,一旦遭到攻击,将会造成巨大的损失[4-6]. 模型检测[7] 是验证智能合约正确性的根本途径. 文献[8] 对 Solidity 智能合约的语法和语义进行了形式化描述,提出了一种可以从 Solidity 程序生成 promela 模型的转换函数,然后使用模型检测工具 SPIN 来验证合约的性质. 文献[9] 提出了一种由不同实体开发和控制的交互式智能合约组成的系统验证方法,使用了模型检测工具 NuSMV 来验证智能合约的功能属性. 这些研究并没有系统地验证智能合约的属性. 并发系统的属性包括安全性和活性[10],智能合约作为一种并发系统,需要满足安全性和活性.

目前已有研究使用模型检测工具 UPPAAL 对智能合约进行验证. 比如文献[11] 提出了一种智能合约的逆向实时模型检测方法,手动地对投票智能合约进行建模并使用 UPPAAL 验证了智能合约的一些性质,但该研究没有具体地给出智能合约转换为时间自动机网络的方法;文献[12] 提出了智能合约的 5 种时间约束模式,使用模型检测工具 UPPAAL 对智能合约带时间约束的性质进行了验证,但该研究仅仅是针对了智能合约带时间约束的性质,没有系统地对智能合约的属性进行验证.

本文基于模型检测技术实现自动化验证,首先定义了 Solidity 基本语句的操作语义及其到时间自动机的转换,将智能合约转换成一个完整的时间自

收稿日期:2022-11-21
基金项目:江西省教育厅科技重点课题(GJJ220302,GJJ210307,GJJ2200303,GJJ2200304)资助项目.
通信作者:王昌晶(1977—),男,江西丰城人,教授,博士,博士生导师,主要从事可信软件、智能化软件与智能化教育的研究. E-mail:wcj@jxnu.edu.cn
卢家兴(1976—),男,江西九江人,副教授,主要从事软件形式化方法、模型检测方面的研究. E-mail:003484@jxnu.edu.cn

动机网络;然后系统地描述智能合约的安全性和活性,并使用模型检测工具 UPPAAL 验证智能合约的属性;最后对购物智能合约进行建模与验证,验证了该方法的有效性.

1 智能合约

文献[13]使用 Solidity 语言编写以太坊中的智能合约,该智能合约先通过编译器编译成字节码,然后在以太坊虚拟机(EVM)上运行[14]. Solidity 智能合约的主题内容由 contract 开始,在 contract 声明中主要包括变量、结构体、事件和函数. 其中变量包括全局变量和局部变量;结构体由关键字 struct 声明;事件是由 EVM 提供的一个便利接口,由关键字 event声明;函数由关键字 function 声明,函数描述的是智能合约具体的功能,是智能合约的核心.

一个智能合约的函数可以表示为

function functionName ([datatype dataName]) external/internal [payable] [modifierName( )] [returns(datatype)]

{…},

其中 datatype dataName 是函数的输入,returns 是函数的输出,若函数无输入输出,则该参数可省略. external/internal 为函数的可见性,external 函数可以被其他的合约调用,而 internal 函数只能在本合约内部运行. payable 表示允许该函数接收以太币. modiferName( ) 是函数的修饰器,可以改变函数的行为. 下面是一个简单的拍卖智能合约,定义了构造函数 constructor( ) 和竞拍函数 bid( ).

```
pragma solidity ^0.8.4;
contract SimpleAuction {
address payable public beneficiary;
//受益人地址
uint public auctionEndTime; //结束时间
address public highestBidder; //最高竞价人
uint public highestBid; //最高竞拍价格
mapping( address⩾uint) pendingReturns;
constructor( uint biddingTime,address payable beneficiaryAddress) { //构造函数
beneficiary = beneficiaryAddress;
auctionEndTime = block.timestamp + biddingTime;}
function bid( ) external payable { //竞拍函数
require( block.timestamp > auctionEndTime);
require( msg.value⩽highestBid)
if ( highestBid ! =0)
pendingReturns [highestBidder] + = highestBid;
highestBidder = msg.sender;
highestBid = msg.value;}}.
```

2 时间自动机网络模型

2.1 模型检测工具 UPPAAL

UPPAAL[15]是一种基于时间自动机的模型检测工具,可以有效地对实时系统建模和自动验证,特别适合对实时系统的安全性和有界活性进行自动验证. 此外,它还可以用于算法分析和协议验证等. UPPAAL 采用一组带有整型时钟变量的时间自动机对实时系统的行为进行建模,用分支时序逻辑(computing tree logic,CTL)来描述系统的性质,使用其工具可对性质进行验证.

UPPAAL 验证器使用 CTL 语言来描述系统的性质,包括安全性和活性. 其 BNF 表示语法为

Prop::$A\,[\!]\,p \mid A\,\langle\rangle p \mid E\,[\!]\,p \mid E\,\langle\rangle p \mid p$ imply q,

其中 $A\,[\!]\,p$ 表示在所有的路径上所有的状态都满足 p;$A\,\langle\rangle p$ 表示在所有路径上未来都有 1 个状态满足 p;$E\,[\!]\,p$表示至少存在 1 条路径,在该路径上所有的状态都满足 p;$E\,\langle\rangle p$ 表示至少存在 1 条路径,该路径未来都有 1 个状态满足 p;p imply q 表示若系统满足 p,则它也满足 q.

2.2 时间自动机

定义 1 一个时间自动机可以定义为一个 6 元组$(L,l_0,\sum,C,I,E)$. 其中 L 是所有位置的集合;l_0 是初始位置;$\sum$ 是迁移的动作集;C 是时钟的集合,$B(C)$ 是时钟约束的集合;I 是每一个位置上的时钟约束不变式;$E\subseteq L\times\sum\times B(C)\times 2^C\times L$ 是有向边的集合.

定义 2 一个时间自动机的语义模型是一个迁移系统 $T=\langle S,s_0,\rightarrow\rangle$. $S\subseteq L\times Z^C$ 表示所有状态的集合;$s_0=(l_0,u_0)$ 表示初始状态;$\rightarrow\subseteq S\times\{Z\cup\sum\}\times S$ 表示迁移关系,其中 $Z\geqslant 0$.

定义 3 一个由多个时间自动机构成的网络可以表示为 $A_i=(L_i,l_i^0,\sum,C,I_i,E_i)$. 设 $\bar{l}_0=(l_1^0,l_2^0,\cdots,l_n^0)$ 为时间自动机网络的初始位置向量,$I(\bar{l})=\bigwedge_{i=1}^{n}I_i(l_i)$ 为组合位置的公共不变量,该时间自动机网络的语义模型是一个迁移系统 $T=\langle S,s_0,\rightarrow\rangle$. $S\subseteq(L_1\times L_2\times\cdots\times L_n)\times Z^C$ 是状态集;$s_0=(\bar{l}_0,u_0)$ 是初始状态;$\rightarrow\subseteq S\times S$ 表示迁移关系.

图 1 展示了一个台灯系统的时间自动机网络. 左边的时间自动机表示台灯时间自动机 Lamp,描述了台灯的行为;右边表示用户时间自动机 User,描述了用户的行为. 台灯共有 3 种状态:off(关闭)、low

(弱亮)、bright(强亮),台灯的初始状态为 off; 用户只有 1 个 Idle 状态; 2 个时间自动机之间通过一个信道 press 来进行同步通信. 用户可以随时按下开关(press!),台灯收到按下开关的信号(press?)时将状态变为 low,并且将时钟 x 置为 0; 当台灯的状态为 low,且用户在 5 个时间单位内($x<5$)再按 1 次开关时,则台灯将会处于 bright 状态; 如果用户超过 5 个时间单位($x\geqslant 5$)按下开关,则台灯的状态将会变为 off. 当台灯的状态为 bright 时,如果用户按下开关,则台灯的状态将变为 off; 如果用户在 50 个单位时间内($x<50$)都没有按下开关,则台灯将会一直处于 bright 状态; 如果用户超过 50 个单位时间都没有按下开关,则台灯状态将会变为 off.

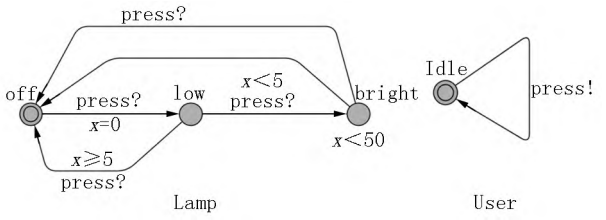
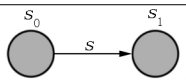


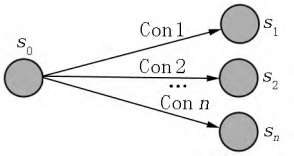
图 1　台灯系统的时间自动机模型

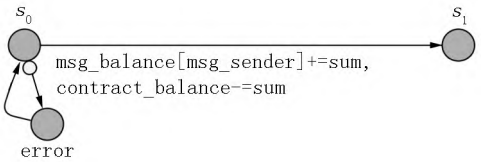
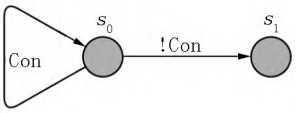
2.3　智能合约的语义及其转换

在 Solidity 中常见的几种语句有赋值语句、条件语句、transfer 语句和循环语句. 为确保智能合约可以完全正确地转换为时间自动机网络模型,对 Solidity 几种语句进行形式化定义. 表 1 给出了在转换后的时间自动机模型.

表 1　Solidity 基本语句的操作语义及转换

Solidity 语句	操作语义	时间自动机
赋值语句	$\dfrac{s}{\langle \text{do\_}S,s\rangle \rightarrow s}$	

条件语句:
if(Con 1) {}
…
if(Con n) {}
	$\dfrac{\langle \text{Con},s\rangle \Rightarrow \text{true}}{\langle \text{if\_Con\_do\_}S,s\rangle \rightarrow \langle S;\ \text{if\_}B\text{\_do\_}S,s\rangle}$
	$\dfrac{\langle \text{Con},s\rangle \Rightarrow \text{false}}{\langle \text{if\_Con\_do\_}S,s\rangle \rightarrow \text{s}}$
		

transfer 语句: msg. sender. transfer(sum)	$\dfrac{(\text{transter},s) \Rightarrow \text{true}}{\langle \text{do\_}S,s\rangle \rightarrow \text{s}}$ $\dfrac{\langle \text{transfer},s\rangle \Rightarrow \text{false}}{s \rightarrow s}$	
循环语句: while(Con) {S}	$\dfrac{\langle \text{Con},s\rangle \Rightarrow \text{ture}}{\langle \text{while\_Con\_do\_}S,s\rangle \rightarrow \langle S;\ \text{while\_}B\text{\_do\_}S,s\rangle}$ $\dfrac{\langle \text{Con},s\rangle \Rightarrow \text{false}}{\langle \text{while\_Con\_do\_}S,s\rangle \rightarrow s}$	

3　智能合约的属性

在并发系统中,一个属性被看作是执行的集合,其包含的元素是满足该属性的执行[16]. 一个智能合约可以看成一个并发系统,如果智能合约系统满足一个属性,那么该智能合约的所有执行都包含在属性所描述的系统中.

定义 4(执行)　执行是由无限或有限多个状态组成的序列,是智能合约 1 次运行. 系统的 1 次运行可以表示为 $\sigma \equiv s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \cdots$. 其中 s_0 表示系统的初始状态,a_i 表示执行的动作.

3.1　安全性

智能合约的安全性是指“坏的事情永远不会发生”. 如果属性 P 是智能合约的一个安全性,那么当智能合约的执行不满足属性 P 的约束时,则说明智能合约在执行时某些状态出现了“坏的事情”,即智能合约不满足安全属性 P.

定义 5(安全性)　属性 P 是智能合约的安全属性,当且仅当对任意不满足属性 P 的执行,每个执行均包含一个有穷前缀 τ,τ 的所有无穷扩展均不满足 P.

智能合约的所有安全属性 P 必然是形如 $\square\psi$. 若一个智能合约某次执行满足属性 P 但不满足 $\square\psi$,则必然存在某个状态序列不满足 P. 本文使用 CTL 公式描述智能合约的属性,利用模型检测工具 UPPAAL 进行验证. 在智能合约中的安全属性包括无死锁性、变量不变性和部分正确性等. 下面以公开拍卖合约[17](简称拍卖合约)为例,介绍智能合约常见的 3 种安全属性.

1) 无死锁性: 指由智能合约转换的时间自动机之间可以正常通信,不会出现“堵死”情况.

在 UPPAAL 中描述为 $A[]$ not deadlock.

2) 变量不变性: 指在智能合约的执行中,某些变量保持不变,或者某些变量之间存在某种恒定的关系. 如在拍卖合约中,任何时候合约账户的余额 contract_balance 与外部账户的余额 account_balance 的和是恒定不变的,假设所有账户初始值的总和为 sum.

在 UPPAAL 中描述为 $A[]$ sum(i: id) account_balance(i) + contract_balance == sum.

3) 部分正确性: 指智能合约执行结束时,所有的结果都正确. 在拍卖合约中,当拍卖结束时,合约账户的余额 contract_balance 为 0.

在 UPPAAL 中描述为 $A[]$ (time > auctionEnd) imply contract_balance == 0.

3.2 活性

智能合约的活性是指“好的事情终会发生”. 活性要求智能合约在执行过程中随时可能发生“好的事情”,即使现在没有发生“好的事情”,将来必定会发生.

定义 6(活性) 属性 P 是智能合约的活性,当且仅当在智能合约中的任意有限状态序列都可以被扩展为一个或多个满足 P 的无限序列.

智能合约的活性通常用公式 $\Diamond\psi$ 表示. 在智能合约中的活性包括终止性、活动性和公平性等. 下面分别描述这几种活性.

1) 终止性: 指在智能合约中的所有执行都能正确地终止. 在拍卖合约中,拍卖总会结束.

在 UPPAAL 中描述为 $A\langle\rangle$AuctionEnd. ended.

2) 活动性: 指在智能合约执行过程中处于某个位置后必定会离开该位置. 在拍卖合约中,用户不会一直处于拍卖的状态.

在 UPPAAL 中描述为 $A\langle\rangle$not Bid. bided.

3) 公平性: 指在智能合约在执行的过程中,必然会到达它可以到达的位置. 在拍卖合约中,失败的用户最终都能获得退款.

在 UPPAAL 中描述为 $A\langle\rangle$PendingReturns[msg_sender] == 0.

4 案例研究

4.1 合约描述

购物合约是一个常见的智能合约. 合约的主要内容是: 管理员添加商家的信息并将商家加入白名单中,商家可以根据商品的 ID 添加商品,用户根据商品的 ID 购买商品. 表 2 展示了在合约中的主要函数及其含义.

表 2 主要的函数及含义

函数	含义
AddSeller()	管理员添加商家信息
WhitelistAddress()	管理员将商家加入白名单
AddProduct()	商家添加商品信息
BuyContent()	顾客购买商品

4.2 合约建模

根据购物合约的流程,将智能合约的函数转换成对应的时间自动机,并设计了一个系统时间自动机与其他时间自动机进行通信,最后结合 UPPAAL 中的变量声明将所有的时间自动机组合成一个完整的时间自动机网络. 图 2 ~ 图 6 分别为 System、AddSeller、WhitelistAddress、AddProduct、BuyContent 等的时间自动机.

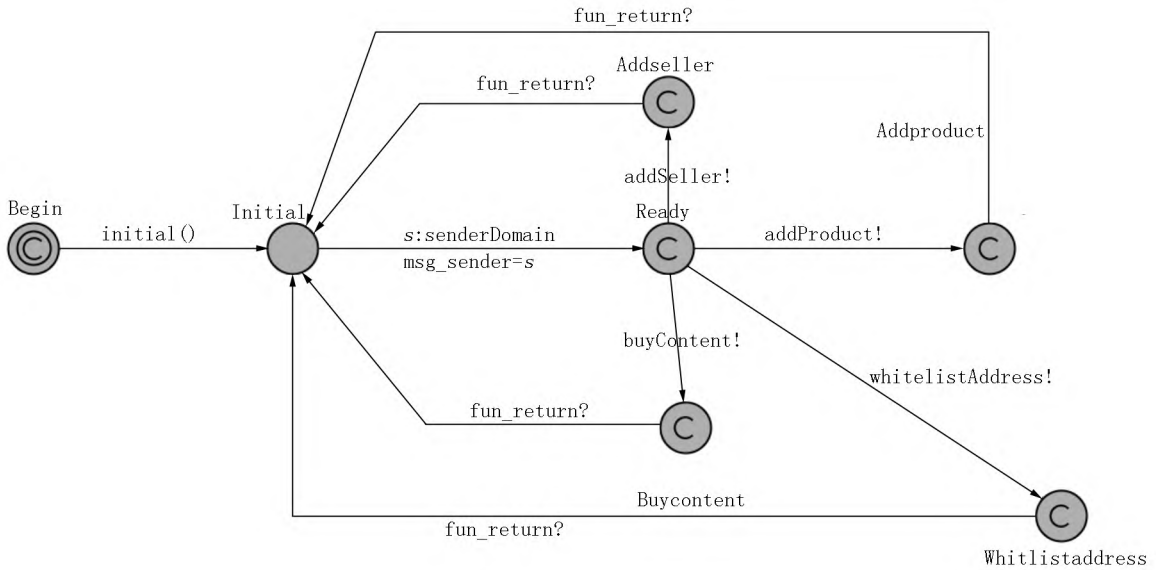


图 2 System 时间自动机

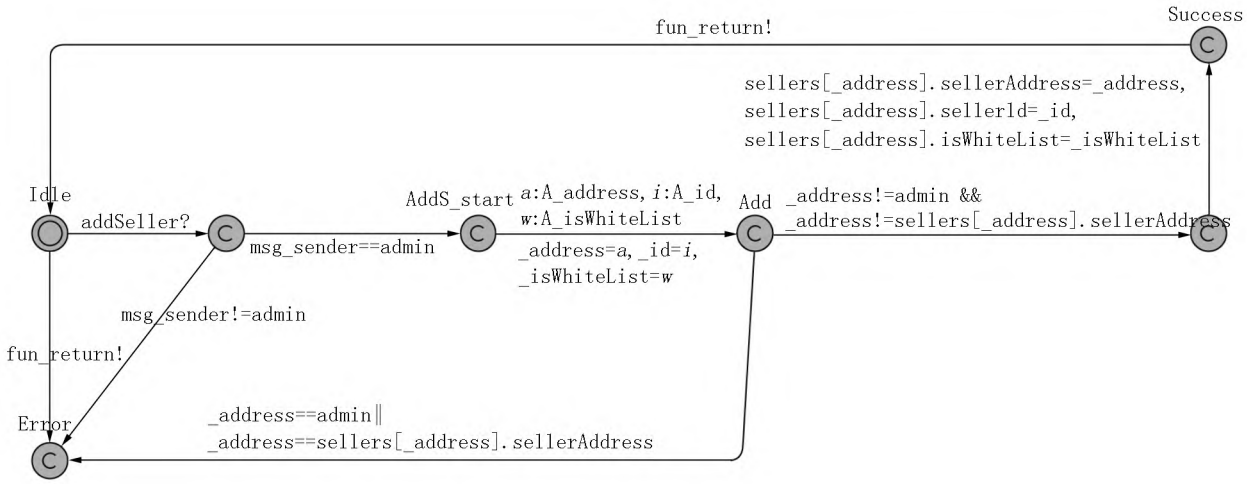


图 3　AddSeller 时间自动机

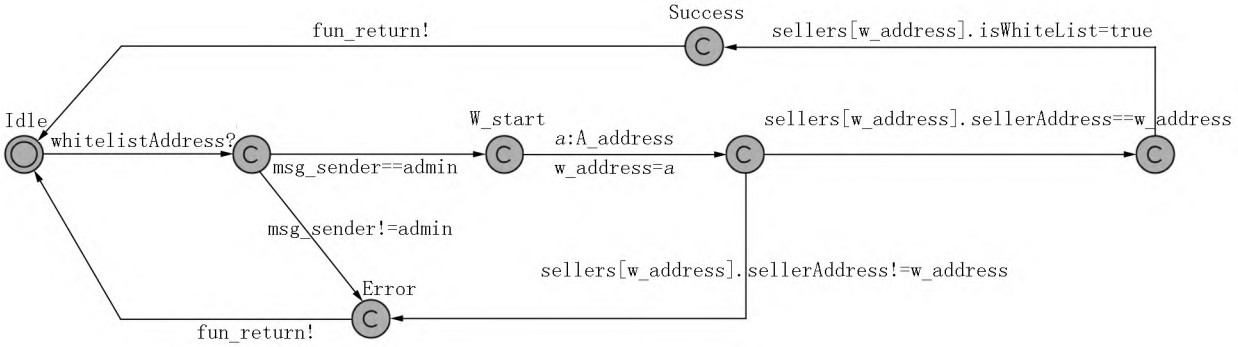


图 4　WhitelistAddress 时间自动机

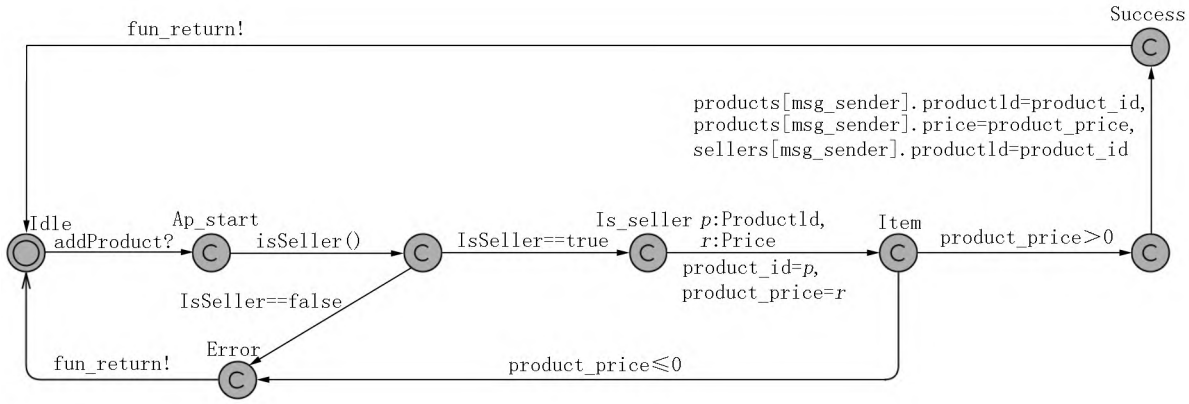


图 5　AddProduct 时间自动机

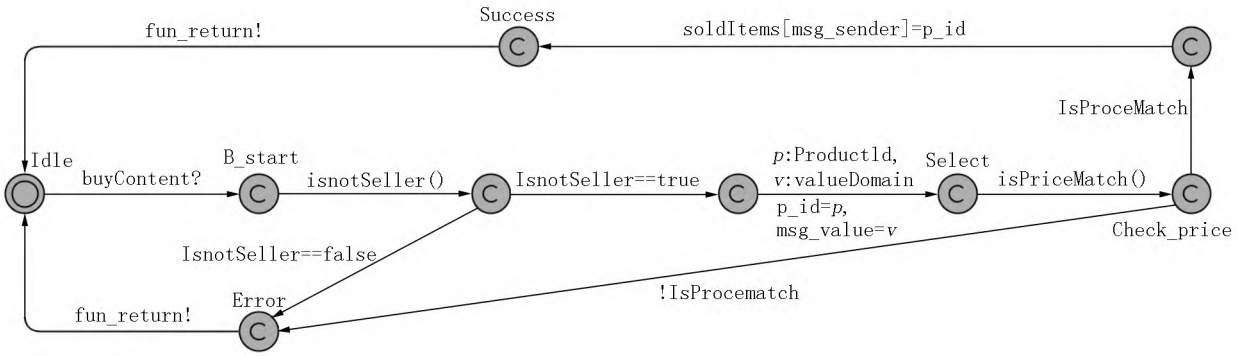


图 6　BuyContent 时间自动机

4.3　属性描述并验证

该节根据第 3 部分的内容用来描述购物智能合约的属性,分别描述了购物合约的安全性和活性. 图 7 是使用 UPPAAL 验证的结果图,表 3 展示了具体

的验证结果.

表 3 验证结果表

属性	验证结果	验证时间/s
无死锁性	满足	116.902
变量不变性	满足	72.318
部分正确性	不满足	0.001
终止性	满足	0.001
活动性	满足	0.021
公平性	满足	0.057

A [] not deadlock
验证费时/kernel 费时/总费时: 116.61 s/0.109 s/116.902 s.
常驻内存/虚拟内存的使用峰值: 460 576 kB/932 964 kB.
满足该性质.
A [] admin = =1
验证费时/kernel 费时/总费时: 72.125 s/0.172 s/72.318 s.
常驻内存/虚拟内存的使用峰值: 460 776 kB/934 884 kB.
满足该性质.
A [] AddProduct.Success imply BuyContent.Success
验证费时/kernel 费时/总费时: 0 s/0 s/0.001 s.
常驻内存/虚拟内存的使用峰值: 460 776 kB/934 884 kB.
不满足该性质.
E <>WhitelistAddress.Success
验证费时/kernel 费时/总费时: 0 s/0 s/0.001 s.
常驻内存/虚拟内存的使用峰值: 460 776 kB/934 884 kB.
满足该性质.
A <>not AddSeller.Add
验证费时/kernel 费时/总费时: 0.187 s/0.031 s/0.21 s.
常驻内存/虚拟内存的使用峰值: 460 776 kB/934 884 kB.
满足该性质.
E <>BuyContent.Success
验证费时/kernel 费时/总费时: 0.063 s/0 s/0.057 s.
常驻内存/虚拟内存的使用峰值: 13 988 kB/36 436 kB.
满足该性质.

图 7 6 种属性验证结果图

属性 1(无死锁性) 时间自动机网络无死锁.

A [] not deadlock.

属性 2(变量不变性) 智能合约运行时管理员的地址是恒定不变的.

A [] admin = =1.

属性 3(部分正确性) 当商品添加成功时,一定会有用户购买.

A [] AddProduct.Success imply BuyContent.Success.

属性 4(终止性) 没有添加白名单的商家最终会被添加到白名单.

E <>WhitelistAddress.Success.

属性 5(活动性) 系统最终会离开添加商家这个位置.

A <>not AddSeller.Add.

属性 6(公平性) 客户最终能成功购买商品.

E <>BuyContent.Success.

验证结果显示,购物合约满足所有的活性,不满足部分正确性这个安全性. 由于安全性是指“坏的事情不会发生”,因此在模型的状态空间中,所有的路径都需要满足安全属性,故验证的时间较长. 而活性是指“好的事情终将发生”,故在模型的状态空间中,只需要 1 条路径满足即可,验证时间较短. 因此在验证过程中,模型更容易满足活性,且验证时间比较短. 对于不满足的性质,系统会给出 1 条不满足的路径,如图 8 所示,给出了不满足部分正确性的路径.

模拟 Trace

(Idle,Begin,Idle,Idle)
System
(Idle,Initial,Idle,Idle,Idle)
System [1]
(Idle,Ready,Idle,Idle,Idle)
addSeller: System→AddSeller
(Idle,Addseller,—,Idle,Idle)
AddSeller
(Idle,Addseller,AddS_start,Idle,Idle)
AddSeller [2,1,0]
(Idle,Addseller,Add,Idle,Idle)
AddSeller
(Idle,Addseller,—,Idle,Idle)
AddSeller
(Idle,Addseller,Success,Idle,Idle)
fun_return: AddSeller→System
(Idle,Initial,Idle,Idle,Idle)
System [2]

System [2]
(Idle,Ready,Idle,Idle,Idle)
addProduct: System→AddProduct
(Ap_start,Addproduct,Idle,Idle,Idle)
AddProduct
(—,Addproduct,Idle,Idle,Idle)
AddProduct
(Is_seller,Addproduct,Idle,Idle,Idle)
AddProduct [2,1]
(Item,Addproduct,Idle,Idle,Idle)
AddProduct
(—,Addproduct,Idle,Idle,Idle)
AddProduct
(Success,Addproduct,Idle,Idle,Idle)

图 8 部分正确性反例路径

5 总结与展望

本文提出了一种智能合约的形式化验证方法. 定义了 Solidity 基本语句的语义及其到 UPPAAL 时间自动机的转换,接着将智能合约的各个功能转换为时间自动机,最后组合成时间自动机网络模型. 描

述了在智能合约中常见的安全性和活性,并使用模型检测工具 UPPAAL 验证智能合约的属性. 通过验证购物智能合约安全性和活性,验证了本文提出的方法的有效性. 本文的实例还处于手工建模阶段,下一步工作将根据智能合约到 UPPAAL 的转换规则,设计出一套完整的转换算法,实现从智能合约到 UPPAAL 时间自动机网络模型的自动转换; 同时也会对智能合约的其他属性进行验证.

6　参考文献

[1] TOLMACH P,LI Yi,LIN Shangwei,et al. A survey of smart contract formal specification and verification [J]. ACM Computing Surveys,2021,54(7):1-38.

[2] SZABO N. Formalizing and securing relationships on public networks [J]. First Monday,1997,2(9):1-21.

[3] DANNEN C. Introducing ethereum and solidity: foundations of cryptocurrency and blockchain programming for beginners [M]. Berkeley: Apress,2017.

[4] MEHAR M I,SHIER C L,GIAMBATTISTA A,et al. Understanding a revolutionary and flawed grand experiment in blockchain: the DAO attack [J]. Journal of Cases on Information Technology,2019,21(1):19-32.

[5] HESSENAUER S. Batch overflow bug on ethereum ERC20 token contracts and safe math [EB/OL]. [2022-09-16]. https://blog.matryx.ai/batch-over flow-bug-on-ethereum-erc20-token-contracts-and-safemath-f9ebcc137434.

[6] 倪远东,张超,殷婷婷. 智能合约安全漏洞研究综述 [J]. 信息安全学报,2020,5(3):78-99.

[7] DEBBI H. Counterexamples in model checking: a survey [J]. Informatica,2018,42(2):145-166.

[8] OSTERLAND T,ROSE T. Model checking smart contracts for Ethereum [J]. Pervasive and Mobile Computing,2020,63:101129.

[9] ALQAHTANI S,He Xinchi,Gamble R,et al. Formal verification of functional requirements for smart contract compositions in supply chain management systems [EB/OL]. [2022-09-22]. https://aisel.aisnet.org/hicss-53/os/blockchain/2/.

[10] 张广泉. 形式化方法导论 [M]. 北京: 清华大学出版社,2015.

[11] 李书霞,王国卿,庄雷. 区块链智能合约安全的逆向实时模型检测方法 [J]. 小型微型计算机系统,2020,41(10):2030-2035.

[12] 赵颖琪,朱雪阳,李广元,等. 智能合约的时间约束模式及其形式化验证 [J]. 软件学报,2022,33(8):2875-2895.

[13] WOHRER M,ZDUN U. Smart contracts: security patterns in the ethereum ecosystem and solidity [EB/OL]. [2022-09-19]. https://ieeexplore.ieee.org/document/8327565/.

[14] VUJI CI C D,JAGODI C D,RAN CI C S. Blockchain technology,bitcoin,and Ethereum: a brief overview [EB/OL]. [2022-09-13]. https://ieeexplore.ieee.org/document/8345547.

[15] BENGTSSON J,LARSEN K,LARSSON F,et al. UPPAAL: a tool suite for automatic verification of real-time system [EB/OL]. [2022-09-19]. https://brics.dk/RS/96/58/BRICS-RS-96-58.pdf.

[16] CHAUDHURI K,DOLIGEZ D,LAMPORT L,et al. Verifying safety properties with the TLA + proof system [EB/OL]. [2022-09-19]. https://members.loria.fr/SMerz/papers/ijcar2010.pdf.

[17] CHRISTIAN R. Simple open auction [EB/OL]. [2022-09-23]. https://learnblockchain.cn/docs/solidity/solidity-by-example.html.

The Formal Verification Method for Smart Contract Properties Based on UPPAAL

ZHANG Qufa[1],WANG Changjing[1,2*],ZUO Zhengkang[1],LU Jiaxing[1*],LIAO Yunyan[1],WANG Yuan[3]

(1. School of Computer and Information Engineering,Jiangxi Normal University,Nanchang Jiangxi 330022,China;
2. Management Science and Engineering Research Center,Jiangxi Normal University,Nanchang Jiangxi 330022,China;
3. School of Software,Jiangxi Normal University,Nanchang Jiangxi 330027,China)

Abstract: Aiming at the problem of smart contract property verification,the formal verification method of smart contract properties based on UPPAAL is proposed in this paper. The operational semantics of Solidity basic statements and its transformation to time automata are defined,and smart contracts are transformed into time automata network models. The properties of smart contracts include safety and activity. The common security and activity of smart contracts are defined and described. The model checking tool UPPAAL is used to verify the properties of smart contracts. The shopping contract is modeled and verified,which proves the effectiveness of the proposed method.

Key words: smart contracts; UPPAAL; time automata; safety; activity　　(责任编辑: 冉小晓)