(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 114510414 B
(45)授权公告日 2024.05.17

(21)申请号 202210047615.4

(22)申请日 2022.01.17

(65)同一申请的已公布的文献号
申请公布号 CN 114510414 A

(43)申请公布日 2022.05.17

(73)专利权人 北京大学
地址 100871 北京市海淀区颐和园路5号

(72)发明人 文伟平 肖遥 刘宇航 胡叶舟 刘军杰 方莹

(74)专利代理机构 北京万象新悦知识产权代理有限公司 11360
专利代理师 李稚婷

(51)Int.Cl.
G06F 11/36(2006.01)

(56)对比文件
CN 108985073 A,2018.12.11
CN 110515856 A,2019.11.29
CN 113672515 A,2021.11.19
CN 113688403 A,2021.11.23
US 2004107174 A1,2004.06.03
US 2004230407 A1,2004.11.18
US 2015213167 A1,2015.07.30

审查员 翟燕燕

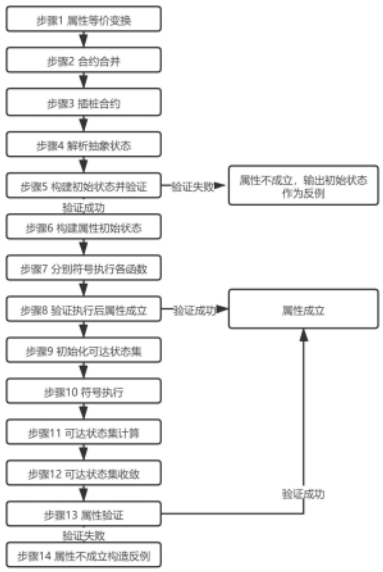
权利要求书2页 说明书6页 附图2页

(54)发明名称

一种基于符号执行对智能合约功能属性进行形式化验证的方法及系统

(57)摘要

本发明公开了一种基于符号执行对智能合约功能属性进行形式化验证的方法及系统,通过合约属性解析将自定义功能属性的证明转化为可达性验证,提供一种利用符号执行自动化进行形式化验证的新途径,对属性断言进行等价转换尝试进行归纳证明能对可归纳性属性达到快速验证的效果,有效提高验证速度,对于不满足的功能属性能够构造对应的反例提高可用性,另外,将符号执行与抽象验证分离开,能够避免不必要的抽象降低符号执行的效率,保证了符号执行收敛的速度。本发明可以自动化对智能合约功能属性进行形式化验证并输出不符合的属性的反例,并通过对合约属性的预处理,可归纳性验证和迭代不动点法相结合有效提高符号执行可达性验证的效率。



CN 114510414 B

1.一种基于符号执行对智能合约功能属性进行形式化验证的方法,首先将自定义功能属性形式化为属性断言,然后抽象该属性断言通过符号执行尝试可归纳性验证,若其具有可归纳性,则可直接证明;若其不具有可归纳性,则通过深度的符号执行不断扩充可达抽象状态集,迭代计算可达抽象状态的不动点,最终从不动点去尝试推导属性断言,若均能成功推导,则已证明了该属性断言,否则,尝试构造反例输出;具体包括如下步骤:

1)接收一个或多个合约C_1,$C_2\cdots C_m$以及输入一系列自定义属性断言φ_1,$\varphi_2\cdots\varphi_n$,将属性断言进行等价变换,变换为$\square\varphi_1'$,$\square\varphi_2'\cdots\square\varphi_n'$的形式,其中$\square\varphi_i'$表示属性断言$\varphi_i'$恒成立,m、n、i代表自然数;

2)将输入的一系列合约合并为一个合约C_0;

3)根据属性断言在合约相应的位置添加新属性变量,插入监测代码,得到插桩合约C_φ,其包含函数$\{f_1,f_2\cdots f_k\}$,其中k代表自然数;

4)根据插桩合约C_φ及等价变换后的属性断言$\square\varphi_1'$,$\square\varphi_2'\cdots\square\varphi_n'$解析所需的抽象状态α;

5)对于一条需要验证的属性断言φ',通过init来构建初始化的抽象状态α_{init},检查在α_{init}下属性断言是否成立,若成立则进入步骤6,不成立则插桩合约C_φ不满足属性断言φ',即证明了输入合约C_1,$C_2\cdots C_m$不满足属性断言φ;

6)将初始化的抽象状态α_{init}转换为初始符号状态集$S_0\{s_{10},s_{20}\cdots s_{m0}\}$;

7)以初始符号状态集中各状态为初始状态,执行函数f_i,得到符号状态集$S_i\{s_{1i},s_{2i}\cdots s_{mi}\}$,并转化为抽象状态集$A_i\{\alpha_{1i},\alpha_{2i}\cdots\alpha_{mi}\}$;

8)若对于某属性断言φ'生成的初始符号状态集S_0分别执行$\{f_1,f_2\cdots f_k\}$得到的符号状态集S_i及其对应的抽象状态集A_i,若φ'均仍然成立,则φ'是可归纳证明属性断言且φ'恒成立,即证明了输入合约C_1,$C_2\cdots C_m$满足属性断言φ;若不成立,则进入步骤9;

9)根据步骤5)中构建的初始化的抽象状态α_{init},初始化可达抽象状态集合$reach(C_\varphi)_0=\{\alpha_{init}\}$;

10)以符号方式执行一次while循环,调用f_i,并在交易结束后计算精确的符号状态s,s实际上是不同路径约束p1∪p2∪…∪p_k的所有可能,包括s_1,$s_2\cdots s_k$;

11)根据精确的符号状态s_1,$s_2\cdots s_k$计算出抽象状态α_1,$\alpha_2\cdots\alpha_m$,由于不同的符号状态可能得到相同的抽象状态,故有k≥m;

12)将抽象状态α_1,$\alpha_2\cdots\alpha_m$加入可达抽象状态集合$reach(C_\varphi)_0$得到可达抽象状态集合$reach(C_\varphi)_1$,重复步骤10)和11);由于抽象状态域是有限的,故最终$reach(C_\varphi)$一定会达到不动点$reach(C_\varphi)_{i+1}=reach(C_\varphi)_i$;

13)根据可达抽象状态集不定点$reach(C_\varphi)_{fix}$验证是否满足属性断言φ',若满足则证明此合约C_φ满足属性断言φ',等价于证明了输入合约C_1,$C_2\cdots C_m$满足属性断言φ,若不满足则进入步骤14);

14)尝试根据该抽象状态对应的符号状态回查已记录的符号执行的路径信息,尝试构建一个反例$\{(user,func,args)_1,(user,func,args)_2\cdots(user,func,args)_n\}$并输出,证明了输入合约$C_1$,$C_2\cdots C_m$不满足属性断言φ。

2.如权利要求1所述的方法,其特征在于,在步骤2)中,多个合约文件互相引用导入的,根据import导入关系逐个合并,另外,为防止冲突,变量名加上合约文件名前缀。

3.一种基于符号执行对智能合约功能属性进行形式化验证的系统,用于实现权利要求1或2所述的基于符号执行对智能合约功能属性进行形式化验证的方法,包括合约属性解析子系统、符号执行子系统、抽象验证子系统;其中:

合约属性解析子系统包括合约合并模块和属性解析模块,合约合并模块用于将多个合约合并为一个合约;属性解析模块用于将属性断言进行等价转换,并根据需要为合约添加新的属性变量;

符号执行子系统包括编译器模块、控制流图构建模块、分支执行模块和约束求解模块,编译器模块用于将智能合约源代码编译为字节码;控制流图构建模块根据字节码构建控制流图;分支执行模块是一个改装过的以太坊虚拟机EVM,能够符号化地运行字节码并记录下执行信息;约束求解模块利用Z3求解器求解路径分支约束条件;

抽象验证子系统包括初始状态生成模块、抽象状态转换模块和属性验证模块,初始状态生成模块用于生成符合属性断言的初始抽象状态;抽象状态转换模块用于由符号状态计算出抽象状态或由抽象状态生成符号状态集;属性验证模块用于验证抽象状态是否满足属性断言。

4.如权利要求3所述的系统,其特征在于,所述合约合并模块根据solidity的语法,只修改变量名避免覆盖即可将多个合约直接合并为一个合约。

5.如权利要求3所述的系统,其特征在于,所述属性解析模块将时间属性均转换为always,所述新的属性变量包括数组和、字典和。

一种基于符号执行对智能合约功能属性进行形式化验证的方法及系统

技术领域

[0001] 本发明涉及计算机安全技术领域,尤其涉及一种基于符号执行对智能合约功能属性进行形式化验证的方法及系统。

背景技术

[0002] 智能合约是一种无需中介、自我验证、自动执行合约条款的计算机交易协议。区块链技术去中心化和数据防篡改等特性,解决了智能合约的信任问题,因此使得智能合约更适合在区块链上进行实现。智能合约作为一种能自动执行合约条款的计算机程序,对于正确性及其他属性具有较高要求,需要尽量避免语法和语义错误,以保证用户的财产安全。目前的审计实践通常检测两类问题:1.通用安全错误,例如重入和溢出;2.自定义功能属性,例如“存款总额永远不会超过合约的余额”和“代币总额不会发生变化”。前者主要通过自动化测试的方法检测,后者则需要形式化方法来验证。

[0003] 形式化方法的本质是基于数学的方法来描述目标软件系统属性的一种技术,适用于计算机软硬件系统的规范、开发和验证。形式化验证是使用基于数学变换的静态分析方法来确定程序规范和代码行为,如从协议规约出发验证程序的安全属性等,其是目前安全级别最高的一种验证方法。由于缺少自动化验证的工具,当前的验证工作需要使用重量级的交互式定理证明程序进行,这些需要大量的人工和专业知识,使审计过程既昂贵又耗时,导致开发人员和审计社区很少采用,迄今为止只有少数智能合约项目得到了正式验证。

发明内容

[0004] 为了克服上述现有技术的不足,本发明提供一种基于符号执行对智能合约功能属性进行形式化验证的方法及系统,将自定义功能属性形式化为属性断言之后,可将自定义功能属性的证明转化为可达性检查,进而通过符号执行精确地验证,通过自动化能有效地减轻审计人员的工作量,并提升审计结果的精确度。

[0005] 本发明提供的技术方案是:一种基于符号执行对智能合约功能属性进行形式化验证的方法,首先需要由合约审计者将自定义功能属性形式化为属性断言(属性断言采用与solidity相同的语法形式,并额外添加了时间属性运算符如Once、Always),然后抽象该属性断言通过符号执行尝试可归纳性验证,若其具有可归纳性,则可直接证明;如果其不具有可归纳性,则通过深度的符号执行不断扩充可达抽象状态集,迭代计算可达抽象状态的不动点,最终从不动点去尝试推导属性断言,若均能成功推导,则已证明了该属性断言,否则,尝试构造反例输出。该技术方案主要包括四个阶段:预处理阶段、归纳证明阶段、迭代不动点阶段、属性验证阶段。其中:

[0006] 预处理阶段执行如下步骤:

[0007] 步骤1,接收一个或多个合约C_1,C_2…C_m以及输入一系列自定义属性断言φ_1,φ_2…φ_n,将属性断言进行等价变换,变换为$\square\varphi_1'$,$\square\varphi_2'$…$\square\varphi_n'$的形式,其中$\square\varphi_i'$表示属性

断言 φ_i' 恒成立,m、n、i代表自然数。

[0008] 步骤2,将输入的一系列合约合并为一个合约C_0。

[0009] 步骤3,根据属性断言在合约相应的位置添加新属性变量,插入监测代码,得到插桩合约C_φ,其包含函数$\{f_1,f_2\cdots f_k\}$,其中k代表自然数。

[0010] 步骤4,根据插桩合约C_φ及等价变换后的属性断言□ φ_1',□ φ_2'…□ φ_n'解析所需的抽象状态α。

[0011] 归纳证明阶段执行如下步骤:

[0012] 步骤5,不考虑合约内调用其他外部合约,合约将以如下循环形式运行:

[0013]
```
init state
while true:
    (user, func, args) := // 任意值
    exec func(args) as user
```

[0014] 因此首先考虑归纳性验证,即对于一条需要验证的属性断言 φ',通过init来构建初始化的抽象状态α_{init},检查在α_{init}下属性断言是否成立。若成立则进入步骤6,不成立则此插桩合约C_φ不满足属性断言 φ',即证明了输入合约$C_1,C_2\cdots C_m$不满足属性断言 φ。

[0015] 步骤6,将上述初始化的抽象状态α_{init}转换为初始符号状态集$S_0\{s_{10},s_{20}\cdots s_{m0}\}$。

[0016] 步骤7,以初始符号状态集中各状态为初始状态,执行函数f_i,得到符号状态集$S_i\{s_{1i},s_{2i}\cdots s_{mi}\}$,并转化为抽象状态集$A_i\{\alpha_{1i},\alpha_{2i}\cdots\alpha_{mi}\}$。

[0017] 步骤8,若对于某属性断言 φ' 生成的初始符号状态集S_0分别执行$\{f_1,f_2\cdots f_k\}$得到的符号状态集S_i及其对应的抽象状态集A_i,若 φ' 均仍然成立,则 φ' 是可归纳证明属性断言且 φ' 恒成立,即证明了输入合约$C_1,C_2\cdots C_m$满足属性断言 φ。若不成立,则进入步骤9。

[0018] 迭代不动点阶段执行如下步骤:

[0019] 步骤9,根据步骤5中构建的初始化的抽象状态α_{init},初始化可达抽象状态集合$reach(C_\varphi)_0=\{\alpha_{init}\}$。

[0020] 步骤10,以符号方式执行一次while循环,调用f_i,并在交易结束后计算精确的符号状态s,s实际上是不同路径约束$p1\cup p2\cup\cdots\cup p_k$的所有可能,包括$s_1,s_2\cdots s_k$。

[0021] 步骤11,根据精确的符号状态$s_1,s_2\cdots s_k$计算出抽象状态$\alpha_1,\alpha_2\cdots\alpha_m$,由于不同的符号状态可能得到相同的抽象状态,故有$k\geq m$。

[0022] 步骤12,将抽象状态$\alpha_1,\alpha_2\cdots\alpha_m$加入可达抽象状态集合$reach(C_\varphi)_0$得到可达抽象状态集合$reach(C_\varphi)_1$,重复步骤10、11。由于抽象状态域是有限的,故最终$reach(C_\varphi)$一定会达到不动点$reach(C_\varphi)_{i+1}=reach(C_\varphi)_i$。

[0023] 属性验证阶段执行如下步骤:

[0024] 步骤13,根据可达抽象状态集不定点$reach(C_\varphi)_{fix}$验证是否满足属性断言 φ',若满足则证明此合约C_φ满足属性断言 φ',等价于证明了输入合约$C_1,C_2\cdots C_m$满足属性断言 φ,若不满足则进入步骤14。

[0025] 步骤14,尝试根据该抽象状态对应的符号状态回查已记录的符号执行的路径信息,尝试构建一个反例$\{(user,func,args)_1,(user,func,args)_2\cdots(user,func,args)_n\}$并输出,证明了输入合约$C_1,C_2\cdots C_m$不满足属性断言 φ。

[0026] 利用上述基于符号执行对智能合约功能属性进行形式化验证的方法,本发明实现了相应的基于符号执行对智能合约功能属性进行形式化验证的系统,包括:合约属性解析子系统、符号执行子系统、抽象验证子系统;其中:

[0027] 合约属性解析子系统包括合约合并模块和属性解析模块。合约合并模块用于将多个合约合并为一个合约,根据solidity的语法,只需修改变量名避免覆盖即可直接合并;属性解析模块用于将属性断言进行等价转换,将时间属性均转换为always,并根据需要为合约添加新的属性变量,如数组和、字典和等。

[0028] 符号执行子系统包括编译器模块、控制流图构建模块、分支执行模块、约束求解模块。编译器模块用于将智能合约源代码编译为字节码;控制流图构建模块根据字节码构建控制流图;分支执行模块是一个改装过的以太坊虚拟机EVM,能够符号化地运行字节码并记录下执行信息;约束求解模块利用Z3求解器求解路径分支约束条件。

[0029] 抽象验证子系统包括初始状态生成模块、抽象状态转换模块和属性验证模块。初始状态生成模块用于生成符合属性断言的初始抽象状态;抽象状态转换模块用于由符号状态计算出抽象状态或由抽象状态生成符号状态集;属性验证模块用于验证抽象状态是否满足属性断言。

[0030] 本发明的有益效果是:

[0031] 本发明提供一种基于符号执行对智能合约功能属性进行形式化验证的方法及系统,通过合约属性解析将自定义功能属性的证明转化为可达性验证,提供一种利用符号执行自动化进行形式化验证的新途径,对属性断言进行等价转换尝试进行归纳证明能对可归纳性属性达到快速验证的效果,有效提高验证速度,对于不满足的功能属性能够构造对应的反例提高可用性,另外,将符号执行与抽象验证分离开,能够避免不必要的抽象降低符号执行的效率,保证了符号执行收敛的速度。本发明可以自动化对智能合约功能属性进行形式化验证并输出不符合的属性的反例,并通过对合约属性的预处理,可归纳性验证和迭代不动点法相结合有效提高符号执行可达性验证的效率。

附图说明

[0032] 图1为本发明基于符号执行对智能合约功能属性进行形式化验证方法的流程框图。

[0033] 图2为本发明实施例提供的基于符号执行对智能合约功能属性进行形式化验证系统的结构框图。

具体实施方式

[0034] 下面结合附图,通过实施例进一步描述本发明,但不以任何方式限制本发明的范围。

[0035] 本发明提供一种基于符号执行对智能合约功能属性进行形式化验证的方法及系统,通过合约属性解析将自定义功能属性的证明转化为可达性验证,提供一种利用符号执行自动化进行形式化验证的新途径,对属性断言进行等价转换尝试进行归纳证明能对可归纳性属性达到快速验证的效果,有效提高验证速度,对于不满足的功能属性能够构造对应的反例提高可用性,另外,将符号执行与抽象验证分离开,能够避免不必要的抽象降低符号

执行的效率,保证了符号执行收敛的速度。

[0036] 图1为本发明验证方法的流程框图,具体实施包括四个阶段:

[0037] 阶段一、对合约及属性进行预处理;

[0038] 阶段二、尝试对属性进行归纳证明;

[0039] 阶段三、利用符号执行迭代不动点获取可达状态集;

[0040] 阶段四、根据可达状态集做属性验证或构造反例。

[0041] 其中阶段一包括如下步骤:

[0042] 步骤1,接收一个或多个合约C_1,C_2…C_m以及输入一系列自定义属性断言φ_1,φ_2…φ_n,将属性断言进行等价变换,变换为$\square\varphi_1'$,$\square\varphi_2'$…$\square\varphi_n'$的形式,$\square\varphi_i'$表示属性断言φ_i'恒成立。

[0043] 自定义属性断言采用一种类似于solidity的语法形式,例如always(sum(DogCoin.balances.value)==10000)表示DogCoin.sol合约中的mapping balances的value之和恒等于10000。

[0044] 非always的属性断言会被等价转换,例如when mintSwitch=false,sum(CatCoin.balances.value)==const表示CatCoin.sol合约中的mapping balances的value之和不变会被转换为always(mintSwitch||(!mintSwitch&&sum(CatCoin.balances.value)==const))。

[0045] 步骤2,将输入的一系列合约合并为一个合约C_0,以solidity为例,多个合约文件互相引用导入的,根据import导入关系逐个合并,另外,为防止冲突,变量名加上合约文件名前缀。

[0046] 步骤3,如果属性断言中涉及到变量的二级信息,如sum(CatCoin.balances.value)用于表示mapping balances的value之和,则根据属性断言在合约相应的位置添加新属性变量用于表征这一信息,插入监测代码,得到插桩的合约C_φ,其包含函数{f_1,f_2…f_k}。

[0047] 步骤4,根据插桩合约C_φ及等价变换后的属性断言$\square\varphi_1'$,$\square\varphi_2'$…$\square\varphi_n'$解析所需的抽象状态α,α包含属性断言所涉及到的所有变量以及根据其二级信息插入的新属性变量。

[0048] 阶段二包括如下步骤:

[0049] 步骤5,不考虑合约内调用其他外部合约,合约将以如下循环形式运行:

[0050]
```
init state
while true:
    (user, func, args) := // 任意值
    exec func(args) as user
```

[0051] 因此首先考虑归纳性验证,即对于一条需要验证的属性断言φ',通过init即合约的构建代码来构建初始化抽象状态α_{init},检查在α_{init}下属性断言φ'是否成立。若成立则进入步骤6,不成立则此合约C_φ不满足属性断言φ',即证明了输入合约C_1,C_2…C_m不满足属性断言φ。

[0052] 步骤6,将上述初始化的抽象状态α_{init}转换为初始符号状态集S_0{s_{10},s_{20}…s_{m0}}。

[0053] 步骤7,根据合约构建控制流图,并以初始符号状态集中各状态为初始状态,分支执行函数f_i,得到符号状态集$S_i\{s_{1i},s_{2i}\cdots s_{mi}\}$,并转化为抽象状态集$A_i\{\alpha_{1i},\alpha_{2i}\cdots\alpha_{mi}\}$,此阶段与传统的符号执行相同。

[0054] 步骤8,若对于某属性断言φ'生成的初始符号状态集S_0分别执行$\{f_1,f_2\cdots f_k\}$得到的符号状态集S_i及其对应的抽象状态集A_i,φ'均仍然成立,则φ'是可归纳证明属性断言且φ'恒成立,即证明了输入合约$C_1,C_2\cdots C_m$满足属性断言φ。若不成立,则进入步骤9。

[0055] 阶段三包括如下步骤:

[0056] 步骤9,根据步骤5中构建的初始化的抽象状态α_{init},初始化可达抽象状态集合$reach(C_\varphi)_0=\{\alpha_{init}\}$。

[0057] 步骤10,以符号方式执行一次while循环,调用f_i,并在交易结束后计算精确的符号状态s,s实际上是不同路径约束$p1\cup p2\cup\cdots\cup p_k$的所有可能,包括$s_1,s_2\cdots s_k$,此阶段与传统的符号执行相同,在分支执行过程中,还会记录分支路径信息,用于不成立时构建反例。

[0058] 步骤11,根据精确的符号状态$s_1,s_2\cdots s_k$计算出抽象状态$\alpha_1,\alpha_2\cdots\alpha_m$,由于不同的符号状态可能得到相同的抽象状态,故有$k\geqslant m$。

[0059] 步骤12,将抽象状态$\alpha_1,\alpha_2\cdots\alpha_m$加入可达抽象状态集合$reach(C_\varphi)_0$得到可达抽象状态集合$reach(C_\varphi)_1$,重复步骤10、11。由于抽象状态域中所包含的抽象变量是有限的,故最终$reach(C_\varphi)$一定会达到不动点$reach(C_\varphi)_{i+1}=reach(C_\varphi)_i$。

[0060] 阶段四包括如下步骤:

[0061] 步骤13,根据可达抽象状态集不定点$reach(C_\varphi)_{fix}$验证是否满足属性断言φ',若满足则证明此合约C_φ满足属性断言φ',等价于证明了输入合约$C_1,C_2\cdots C_m$满足属性断言φ,若不满足则进入步骤14。

[0062] 步骤14,尝试根据该抽象状态对应的符号状态回查已记录的符号执行的路径信息,尝试构建一个反例$\{(user,func,args)_1,(user,func,args)_2\cdots(user,func,args)_n\}$并输出,证明了输入合约$C_1,C_2\cdots C_m$不满足属性断言φ。

[0063] 利用上述基于符号执行对智能合约功能属性进行形式化验证的方法,本发明实现了相应的基于符号执行对智能合约功能属性进行形式化验证的装置,图2为本实施例提供的智能合约属性形式化验证系统的结构框图,包括如下子系统:

[0064] 子系统一、合约属性解析子系统;

[0065] 子系统二、符号执行子系统;

[0066] 子系统三、抽象验证子系统。

[0067] 其中,子系统一包括如下模块:

[0068] 模块一、合约合并模块,根据import导入层级关系,将多个合约合并为一个合约,另外,给变量名加上前缀避免命名冲突;

[0069] 模块二、属性解析模块,将属性断言进行等价转换,将时间属性均转换为always,并根据需要为合约添加新的属性变量,如数组和、字典和等。

[0070] 子系统二包括如下模块:

[0071] 模块三、编译器模块,根据合约选取相应版本的solidity官方编译器将智能合约源代码编译为字节码,同时保留变量、代码对应信息;

[0072] 模块四、控制流图构建模块,该模块根据字节码构建控制流图;

[0073] 模块五、分支执行模块,这是一个改装过的以太坊虚拟机EVM,能够符号化的运行字节码并记录下执行信息;

[0074] 模块六、约束求解模块,利用Z3求解器求解路径分支约束条件。

[0075] 子系统三包括如下模块:

[0076] 模块七、初始状态生成模块,用于生成符合属性断言的初始抽象状态,具体的需要先从属性断言中解析出具体的约束,生成带有约束的抽象状态;

[0077] 模块八、抽象状态转换模块,能够实现符号状态与抽象状态的相互转化,其中从符号状态到抽象状态可以通过计算得到,抽象状态生成符号状态集则可以通过带约束的符号化来得到;

[0078] 模块九、属性验证模块,通过将属性断言转换为约束条件,可以将属性验证转换为与抽象状态的联合求解,验证抽象状态是否满足属性断言。

[0079] 需要注意的是,公布实施例的目的在于帮助进一步理解本发明,但是本领域的技术人员可以理解:在不脱离本发明及所附权利要求的精神和范围内,各种替换和修改都是可能的。因此,本发明不应局限于实施例所公开的内容,本发明要求保护的范围以权利要求书界定的范围为准。

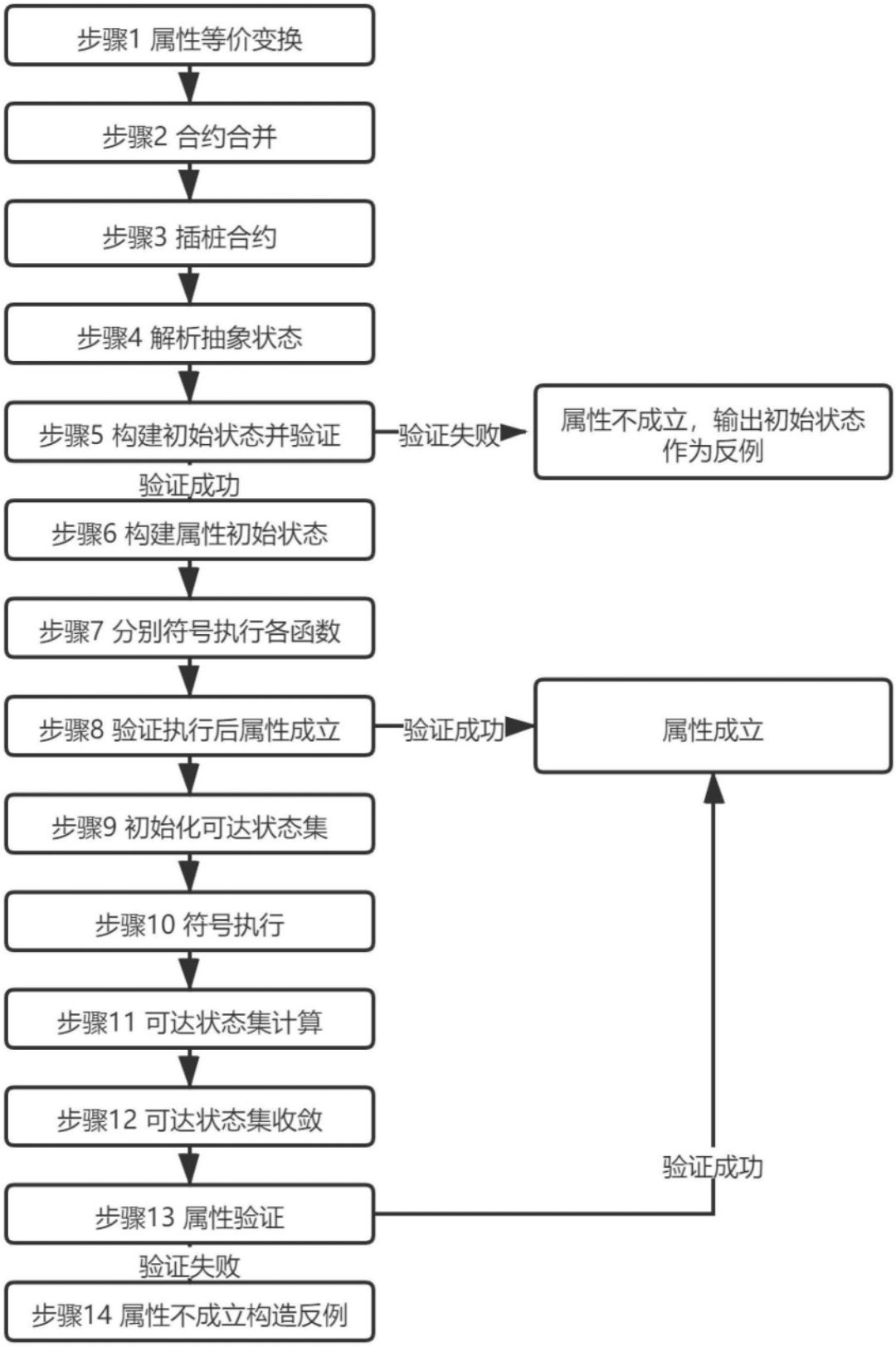

步骤1 属性等价变换
步骤2 合约合并
步骤3 插桩合约
步骤4 解析抽象状态
步骤5 构建初始状态并验证
验证失败
属性不成立，输出初始状态作为反例
验证成功
步骤6 构建属性初始状态
步骤7 分别符号执行各函数
步骤8 验证执行后属性成立
验证成功
属性成立
步骤9 初始化可达状态集
步骤10 符号执行
步骤11 可达状态集计算
步骤12 可达状态集收敛
验证成功
步骤13 属性验证
验证失败
步骤14 属性不成立构造反例

图1

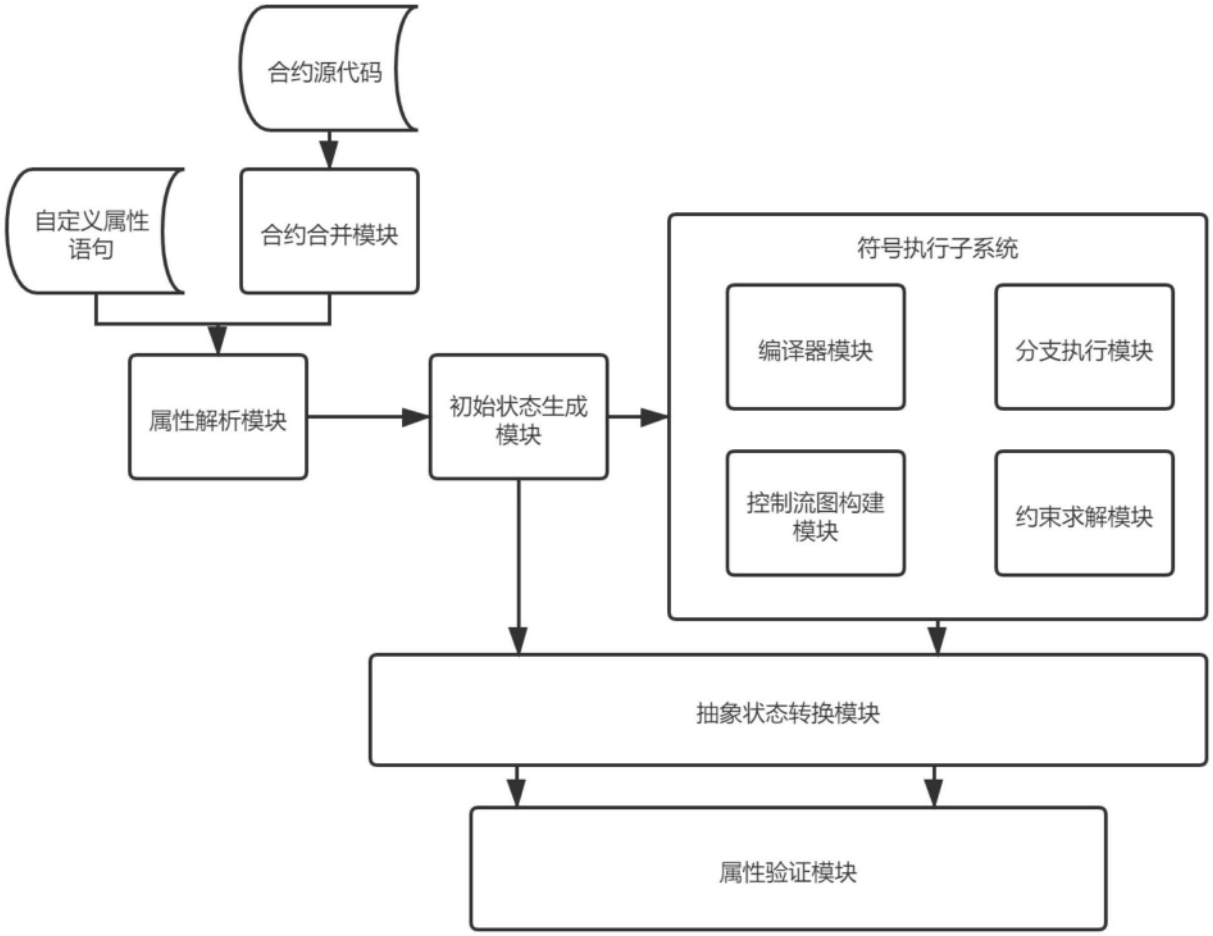

合约源代码
自定义属性语句
合约合并模块
属性解析模块
初始状态生成模块
符号执行子系统
编译器模块
分支执行模块
控制流图构建模块
约束求解模块
抽象状态转换模块
属性验证模块

图2