**面向IFTTT智能家居物联网的安全验证工具**

南京大学大学生创新项目结题报告

**吴秉乐 朱梓源 陈昌繁 陈鹏宇**

**指导老师 卜磊 副教授**

（南京大学 计算机科学与技术系，江苏 南京 210023）

**摘 要：**近年来的各项网络技术之中，区块链与分布式计算倍受关注。以太坊将去中心化货币应用与智能合约相结合，成为了全球最大的分布式应用软件开发平台。由于区块链的特性，搭载于以太坊上的智能合约必须有足够的安全性，否则数字资产风险过大，进而影响整个以太坊生态。由于智能合约对安全性的迫切需求，我们希望借助模糊测试技术，在智能合约安全性测试方面进行一定的探索，尝试开发自动化的测试工具。

**关键词：**区块链；智能合约；模糊测试

一、引言

1.1 选题背景

**1.1.1 区块链简介**

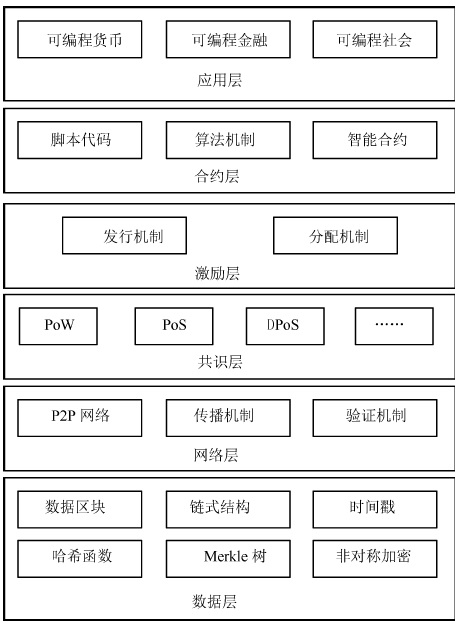
区块链无疑是当前时代下最为炙手可热的技术之一。2008年，第一个区块链由中本聪（Satoshi Nakamoto）《比特币：一种点对点的电子现金系统》一文概念化而出现。2009年1月3日序号0的创世区块诞生，中本聪将它作为加密货币比特币的核心组成部分部署实施，在那里它作为所有交易的公共分类账，比特币（Bitcoin）从此诞生。

区块链是一个分布式的共享账本和数据库，一个区块链包含一串使用密码学方法相关联的数据块（区块），每个数据块中包含一批次的区块链交易信息，用以验证区块链的信息有效性，类似于区块链网络内部所有人各自持有内容相同的账本，交易在所有账本上同时记录。区块链网络内部维护同一串区块的唯一与有效共识，如果某些节点对数据块的数据进行篡改，由于交易的可验证性，这一个有问题的区块将不会被大多数节点共识承认，也就不会影响整体区块链的可信性，类似于一个错误的账本内容只会被社群里其他账本纠错。因其通过分布式数据存储、点对点传输、共识机制、加密算法等技术的集成，具有去中心化、不可篡改、全程留痕、可以追溯、公开透明、集体维护等特征，可以极大的推动数据的可信存储、商业协同、数据可信的交换和分享，以及随之诞生的新兴商业模式，从而构建可信交易环境，打造可信社会。

**1.1.2智能合约与以太坊简介**

在去中心化数字货币的发展中，中本聪的区块链是第一个可靠的去中心化解决办法。现在，开发者们的注意力开始迅速地转向比特币技术的第二部分：区块链怎样应用于货币以外的领域，智能合约（smart contract）应运而生。一个智能合约是一套以数字形式定义的承诺（promises） ，包括合约参与方可以在上面执行这些承诺的协议。智能合约的诞生使得区块链上的交易更为智能和可信，如果说基础区块链是一个可信的记账人的话，运行于区块链之上的智能合约就是一个可信的交易中间人，使得区块链的交易从简单的转账交易跨越到依托自动执行的智能合约代码的类型丰富的交易方式。

通过实现区块链上的智能合约，以太坊（Ethereum）基于脚本、竞争币和链上元协议（on-chain meta-protocol）概念进行整合和提高，使得开发者能够创建任意的基于共识的、可扩展的、标准化的、特性完备的、易于开发的和协同的应用。以太坊通过建立终极的抽象的基础层——内置有图灵完备编程语言的区块链——使得任何人都能够创建合约和去中心化应用，并在其中设立他们自由定义的所有权规则、交易方式和状态转换函数。以太坊因此成为第一个、也是目前最大的开源公共智能合约区块链平台。



区块链架构

此后，一般的区块链系统由数据层、网络层、共识层、激励层、合约层和应用层组成。其中，数据层封装了底层数据区块以及相关的数据加密和时间戳等基础数据和基本算法；网络层则包括分布式组网机制、数据传播机制和数据验证机制等；共识层主要封装网络节点的各类共识算法；激励层将经济因素集成到区块链技术体系中来，主要包括经济激励的发行机制和分配机制等；合约层主要封装各类脚本、算法和智能合约，是区块链可编程特性的基础；应用层则封装了区块链的各种应用场景和案例。

**1.1.3智能合约的安全隐患**

以太坊拥有搭建于区块链上的图灵完备的虚拟机（EVM），可以编写完整的程序。以太坊作为开源的公共区块链平台，其所有用户都可以看到基于区块链的智能合约。这就意味着包括安全漏洞在内的所有漏洞都是可见的。如果智能合约开发者疏忽或者测试不充分，造成智能合约的代码有漏洞的话，就非常容易被黑客利用并攻击。并且越是功能强大的智能合约，就越是逻辑复杂，也越是容易出现逻辑漏洞。同时，智能合约语言Solidity自身与合约设计都有存在漏洞的可能性。

以太坊在发展过程中已经出现过多次重大安全事件。2016年6月17日发生了在区块链历史上沉重的一次攻击事件。由于以太坊的智能合约存在着重大缺陷,区块链业界最大的众筹项目TheDAO遭到攻击，导致300多万以太币资产被分离出TheDAO 资产池。2017年7月21日，智能合约编码公司Parity警告1.5版本及之后的钱包软件存在漏洞，据Etherscan.io的数据确认有价值3000万美元的15万以太币被盗。2017年11月8日，以太坊Parity钱包再出现重大bug，多重签名漏洞被黑客利用，导致上亿美元资金被冻结。

以太坊的智能合约目前已知存在Solidity漏洞、短地址漏洞、交易顺序依赖、时间戳依赖、可重入攻击等漏洞。对于合约调用者，对开源甚至不开源的源码不完全熟悉，不能期待其对合约风险有完整理解，调用他人的智能合约可能会遭到隐含的漏洞的利用；对于合约创建者，由于智能合约部署后难以更新，合约的漏洞可能造成极大的损害。因此，合约安全性的检验是十分必要的。

1.2 项目知识基础和实践基础

**1.2.1 知识基础**

要完成这个应用，首先需要的是扎实的知识基础以及不错的编程能力。在知识方面，项目成员均为计算机与金融工程实验班学生，有一定的计算机知识的基础，且对区块链以及其较大的金融价值有一定的理解。这些知识都为项目的实现打下了基础。

而在技术方面，我们主要考虑应用模糊测试方法对智能合约的安全性进行检测。全体组员在这个方面进行了认真的学习研究。

模糊测试是一种被广泛应用于软件测试中的技术，主要针对软件的安全性检测，其本质是通过随机生成大量的输入数据来识别软件安全漏洞。这种技术的有效性集中体现在当输入数据为边界数据或不合法数据时，能够极具针对性地暴露软件的隐藏漏洞，同时可以使软件工程师能够及时准确的修补漏洞，避免安全隐患。模型检测已被应用于计算机硬件、通信协议、控制系统、安全认证协议等方面的分析与验证中，取得了令人瞩目的成功，并从学术界辐射到了产业界。模糊测试的优点在于这种技术的自动化程度高，只需要测试程序随机或半随机地生成测试用例；可以应用于隔离源代码的黑盒测试模式，适用范围更广；是动态实际执行的，比静态检测更好地反应程序状况、减少误报。

以太坊虚拟机运行智能合约与传统程序的处理器机制类似，调用者输入正确格式的参数发起交易，开始运行智能合约；由以太坊虚拟机执行一系列编译后得到的操作码，进行运算并得到修改链上数据的结果，最终将广播后的交易与结果验证打包于新的区块中。在这个过程中应用模糊测试，可以用不同的测试用例调用智能合约，在以太坊虚拟机执行操作码时插桩分析，最终发现合约中可能存在的漏洞。

**1.2.2 自主兴趣和实践基础**

除了知识和技术基础，兴趣也是必不可少的一项。除了对实现一个完整的应用软件所带来的兴奋感和运用已学知识以及自学的新知识对新的领域进行研究的好奇心之外，由于这个项目是自主选题，课题方向正是组里的各个成员所关心和喜爱的，这为项目的推动带来了巨大的动力。

另一方面，代码开发、问题解决是大家经过完成各种实验和课题所培养出来的能力，大家都有丰富的实践基础。

1.3 项目目的与意义

考虑到以太坊与财产的相关性，以及智能合约部署后难以修改的性质，如果智能合约中有安全相关的漏洞，就很有可能被黑客利用，导致严重后果。在这种情况下，人们就希望能有工具来帮助检测代码，避免合约漏洞而导致的一系列麻烦。

在以太坊智能合约的安全性检测方面应用模糊测试，不仅可以帮助开发者于在公链部署合约之前进行更详尽的测试来及时修正合约漏洞，也可以帮助合约调用者在调用前对被调用的目标合约进行一定程度的测试，以确保此次调用不会产生非预期的恶劣结果。

开展此次项目有助于团队成员参与项目研究全过程，对提升成员的创新实践能力有一定的帮助。研究中对问题的解决过程，也有助于提升团队成员独立思考、学术研究、语言表达、团队协作、心理承受能力。宝贵的创新实践经历将有助于我们团队成员成长为祖国栋梁。

二、项目的创新点与特色

我们的项目有以下创新点与特色点。

2.1 智能分析abi的模糊测试

为达成模糊测试的调用智能合约步骤，我们开发的工具可以在构建测试用例前对智能合约要求的输入abi接口进行分析，进而生成规则内且有足够多样性的测试用例，避免大量测试调用因为输入不符合接口而失败。

2.2 分析调用结果指导用例生成

为提高测试覆盖率，我们开发的工具在插桩时获取了虚拟机储存区的数据，在已知条件分支判断对应数据在这部分数据中的情况下，将这些数据整理后分析，指导测试用例生成，或有助于提升模糊测试覆盖率，从而提升对安全性漏洞检测的成功率。

2.3 适用于多种环境的模糊测试

我们的工具包含了一个以太坊官方发布的Go语言编写的钱包（geth）的修改版本，但修改没有破坏其构建共识的部分。因此我们的工具可以在私有链上进行测试，也可以在暂时性脱机的公有链上测试（脱机避免测试造成财产损失）。

2.4 综述

除了以上的创新点外，在技术和算法上我们的项目组也有一定的特色与创新：

在家具建模中，以应用广泛的json数据格式对智能设备的特性进行抽象。用户可以填充IFTTT指令框架将设备互联。

在模型检验中，以两种方式——Fsm与Lha对系统建模，分别利用目前较成熟的Nusmv以及bach工具进行验证。同时我们设计了自动化修复系统的算法。修复的原理是将设定的Rules中的数值参数化，寻找反例，找到最合适的参数范围，返回一个合适的参数给用户。

在软件设计中，采用了可撰写跨平台应用的Java语言，具有良好的移植性。

三：项目实施的收获与体会

历时一年，本项目基本完成了预期的目标，可以实现用户自定义智能家居系统，能够检测出系统可能的安全隐患，并实现系统的自动化修复。

由于智能设备的快速发展，以IFTTT为代表的设备互联公司的出现，使得用户能够个性化的定义自己的智能系统，然而这样的用户私人定制的产品背后隐藏着巨大的风险。本项目则基于用户定义系统的安全性问题，提出了自动修复隐患的创新性的想法。实现基于反例的自动化修复，规避系统可能出现的安全性问题，保证用户使用系统的安全性。

在项目进行过程中，我们学习到了多方面的知识，如模型检验技术，图形化编程，混成系统等，扩宽了自己的知识面，加深了对自身学科领域的了解。通过与导师两周一次的讨论中，加深了对项目本身的理解，掌握了项目背后的原理。

从团队合作角度来说，大家在项目进行过程中的积极交流也让我们收获良多，无论哪个组员在项目进行过程中遇到新的问题，大家都能尽自己所能想出解决方案。在不断的讨论中，我们对这个项目有了更深刻和清晰的理解，功能实现也更加的完善。

总之，本创新项目是在家具互联日益发展的基础上完成了对互联系统的安全性检测和修复，能够保证用户在使用自己自定义系统时的安全性，对用户的安全保障具有重大意义，对于我们项目组成员来说，增加了团队合作意识，也提升了个人的视野和能力，着实受益匪浅。

最后特别感谢卜磊老师和熊文学长给予我们的指导和帮助！感谢在项目开展过程中所有给予我们帮助支持和鼓励的同学们！

四：项目实施的进展情况及取得的创新成果

4.1 项目实施的进展情况

整个项目的实施，我们的研究过程主要有以下5个阶段

* 前期准备（2015.09~2015.10）具体包括：研读相关书籍和论文，了解形式化验证和模型检验等相关技术；学习java并重点其研究图形化编程的相关知识；学习如何建立自动机模型并了解自动化修复的相关算法。
* 分项研究（2015.11~2016.4）四个组员两两分组，分别负责系统不同模块的具体实现：分别为基于图形界面的仿真系统界面实现和基于模型检验技术的验证和修复功能的实现。
* 功能模块整合及测试（2016.4~2016.7）将两个功能模块进行代码整合，并进行基本功能测试。主要包括：a.能否实现用户自定义家具互联的IFTTT命令;b.能否检测出用户自定义命令可能存在的安全隐患；c.能否自动修复可能隐患
* 功能拓展（2016.8~2016.10）在原有系统建立的有穷自动机（Finite State Machine）模型的基础上，增加对线性混成系统（Linear Hybrid Automata）建模和修复的功能实现，以期能更真实的模拟实时控制系统的行为和发现系统问题。
* 项目总结（2016.11~2016.12）总结项目经验，整理研究成果。

具体两个模块的研究过程如下：

**4.1.1 前端图形化界面**

由于之前未接触过图形化编程，在与导师商讨后，决定以java实现项目界面。故而 在立项后的一段时间内，主要学习和熟悉了java语言的语法语义。相对熟悉地掌握java 的 基本语法和应用后，就开始着手学习java的图形化界面编程，并在学习过程中逐步打起了图 形界面的大体框架，分为“add（添加）”、“Check（检验）”、“Fix（修复）”3个界面，分别用 于家具以及命令的添加、对是否满足规约的检测和自动修复当前系统存在的安全隐患。后来 为了方便用户查看可能存在安全隐患的行为，添加了“Demo（演示）”功能，主要用来向 用户展示可能到达危险状态的行为路径，使得界面对用户而言更加友好和直观。

**4.1.2 后台检验和修复**

本模块的研究过程分为两个部分：实现用户自定义系统的建模，并利用现有成熟的 模型检验工具进行验证；基于验证反例的自动化修复。

4.1.2.1 自动化建模

前期用了较长时间理解模型检验技术的基本原理和已有的相关检验工具，在咨询了 导师的建议后决定采用NuSMV模型检验工具，并通过研究相关manual和tutorial，熟悉 NuSMV的用法，手动建模。

在寒假前后，依据代码整合的需要和NuSMV可运行的系统平台，决定采用java在 Linux平台下实现自动建模。根据NuSMV定义的语法规范，通过读入相关家具的json文 件以及用户设置的IFTTT命令和安全性规约生成SMV文件，调用NuSMV，将程序运行结果 递交给自动化修复模块。

在暑假前后，为了增加对线性混成系统建模的实现，重新设计相关的数据结构，使 其能兼容不同系统的数据存储，因此只需要针对不同系统，根据相关模型检验工具的输 入文件语法规范，生成相对应的模型文件。

4.1.2.2 基于反例的自动化修复

由于系统可供修改的只有用户设定的IFTTT命令，因此本模块创造性地提出了一种基 于验证反例的自动化修复方法：将IFTTT命令条件参数化，利用NuSMV找出一组不满足原规 约的否定的参数值，然后检验该组参数值是否满足之前的规约。由于现实系统中一般不存在 只有离散化的逻辑控制行为，还具有连续性的时间性行为，因此，后期主要研究混成系统的 自动化修复过程。原理同有穷自动机模型，根据用户设定的rule和一组不可达的危险状态， 利用现有工具循环查找合适的参数值，以实现自动化修复。

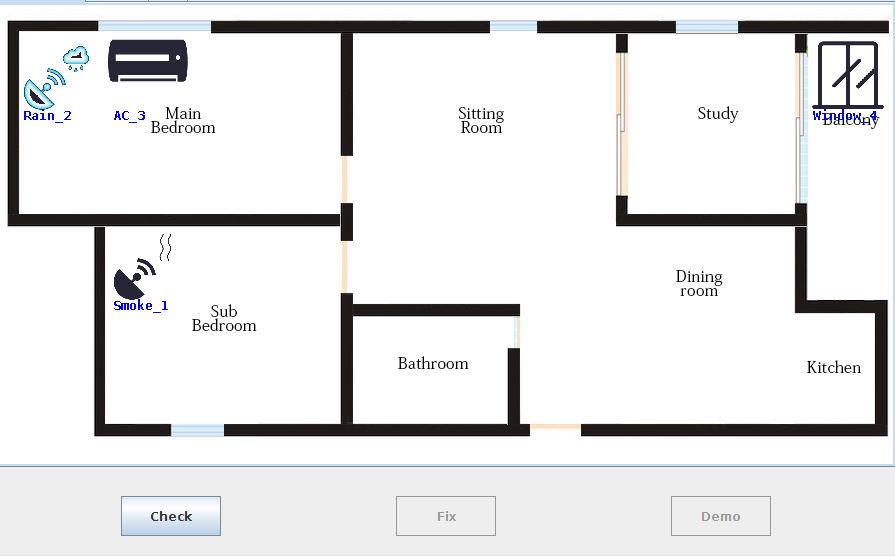
**4.2 取得的创新成果**

经过一年的研究和学习，我们开发出基于Linux平台的智能家居自动仿真系统。其具有 用户自主配置相关设备和IFTTT命令、检验用户配置系统的安全性、修复系统可能出现的安 全隐患等功能。系统能够通过图形界面直观的显示家具之间的规则和安全规约，并能够对可 能出现安全隐患的行为进行模拟，最终实现基于反例的自动化修复，对存在安全隐患或不满 足用户设定规约的规则进行最优化的修正并反馈给用户，基本完成预期成果。

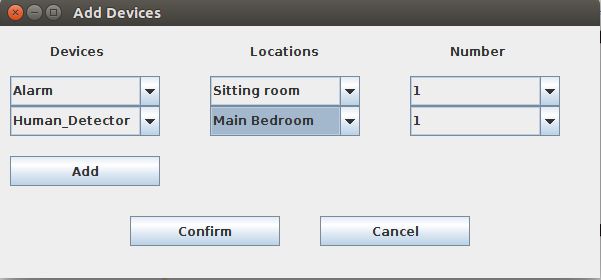
以下将就前段和后台来详细介绍研究成果

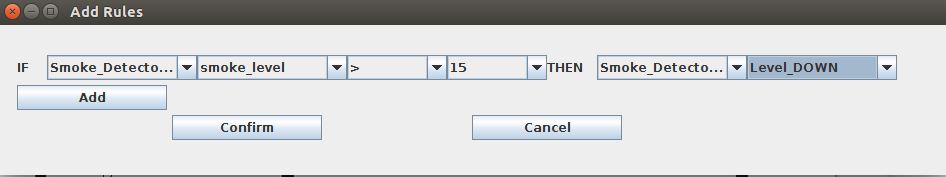
**4.2.1不同用户界面的展示**

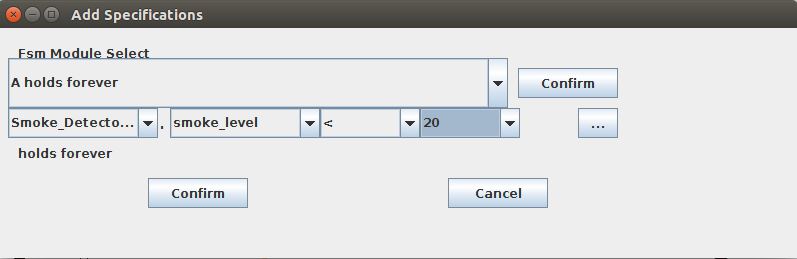
4.2.1.1主界面：包含用户房间的仿真，用户可以自主添加智能家具，并设置IFTTT 规 则对家具进行互联和添加安全性规约，并且可以将保存当前配置，以便下次使用；



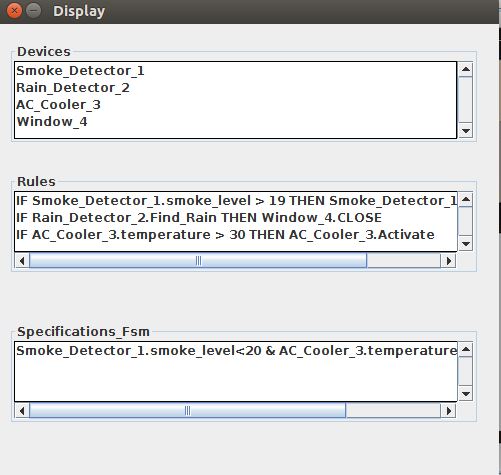
4.2.1.2 添加功能：用户自主添加家具、规则、规约



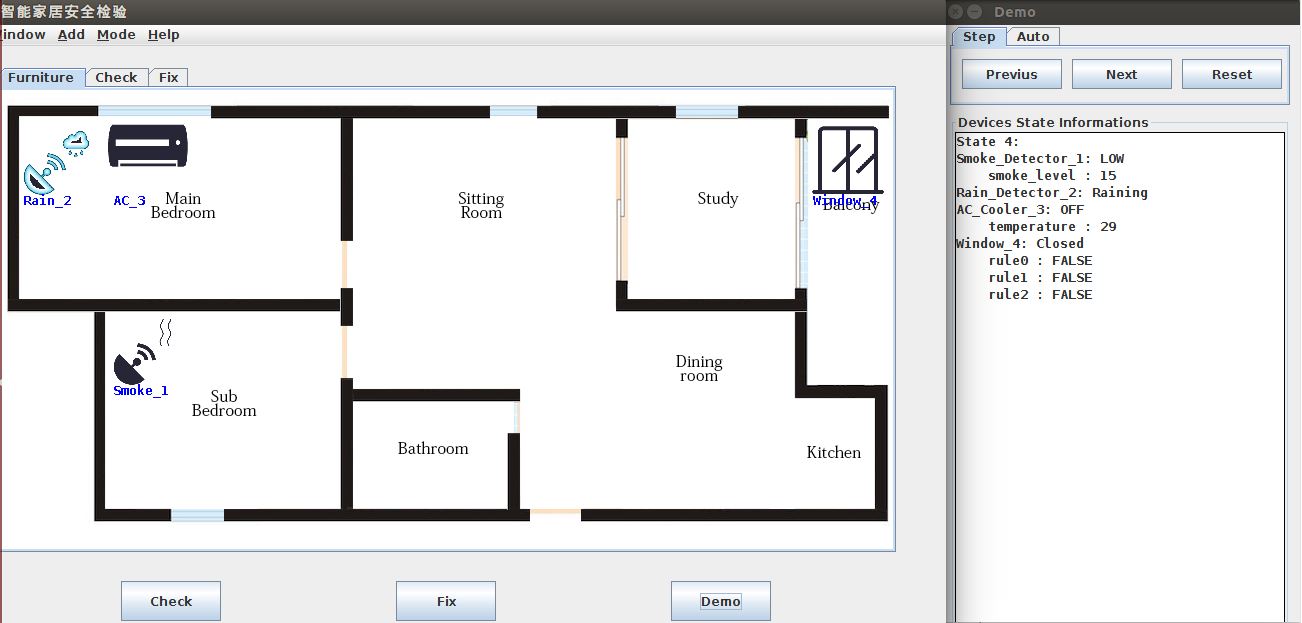




4.2.1.3 查看功能：显示当前家具、规则以及规约，用户可对rule和规约进行修改和删除；

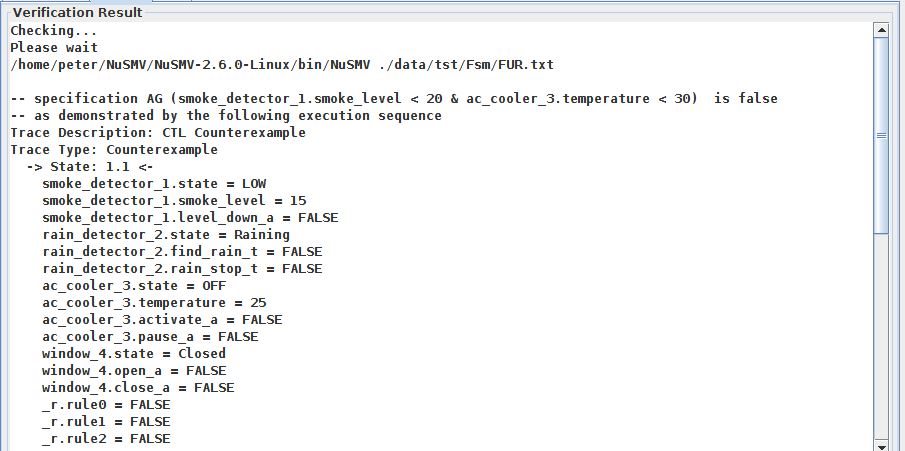


4.2.1.4 Demo：显示到达危险状态的家具行为路径，有穷状态机模型显示家具状态和变量 的离散变化，混成系统模型显示家具状态和变量随时间的连续变化；

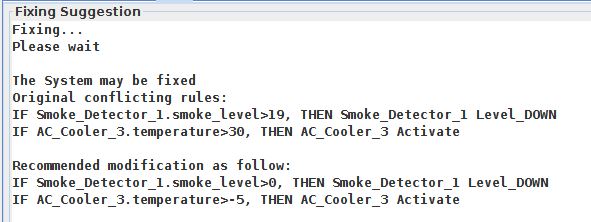


**4.2.2 检测和修复界面**

4.2.2.1 检测模块通过调用模型检验工具来检测系统的安全性，若系统存在安全隐患， 还会显示出一条可能的到达危险状态的路径。



4.2.2.2 修复模块通过在4.1.2.2中介绍的算法对系统进行修复，并向用户展示需要修 改的规则，用户可以点击ok按钮自动修改rule并保存。



五．收获与体会

历时一年，本项目基本完成了预期的目标，可以实现用户自定义智能家居系统，能够检测出系统可能的安全隐患，并实现系统的自动化修复。

由于智能设备的快速发展，以IFTTT为代表的设备互联公司的出现，使得用户能够个性化的定义自己的智能系统，然而这样的用户私人定制的产品背后隐藏着巨大的风险。本项目则基于用户定义系统的安全性问题，提出了自动修复隐患的创新性的想法。实现基于反例的自动化修复，规避系统可能出现的安全性问题，保证用户使用系统的安全性。

在项目进行过程中，我们学习到了多方面的知识，如模型检验技术，图形化编程，混成系统等，扩宽了自己的知识面，加深了对自身学科领域的了解。通过与导师两周一次的讨论中，加深了对项目本身的理解，掌握了项目背后的原理。

从团队合作角度来说，大家在项目进行过程中的积极交流也让我们收获良多，无论哪个组员在项目进行过程中遇到新的问题，大家都能尽自己所能想出解决方案。在不断的讨论中，我们对这个项目有了更深刻和清晰的理解，功能实现也更加的完善。

总之，本创新项目是在家具互联日益发展的基础上完成了对互联系统的安全性检测和修复，能够保证用户在使用自己自定义系统时的安全性，对用户的安全保障具有重大意义，对于我们项目组成员来说，增加了团队合作意识，也提升了个人的视野和能力，着实受益匪浅。

最后特别感谢卜磊老师和熊文学长给予我们的指导和帮助！感谢在项目开展过程中所有给予我们帮助支持和鼓励的同学们！

References:

1. Jr. Clarke EM, Grumberg O, Peled DA. Model Checking. Cambridge: The MIT Press, 1999.
2. Hall, A.: Seven myths of formal methods. IEEE Softw. 7(5), 11–19 (1990)
3. JML Specification Language. <http://www.jmlspecs.org>
4. Bu Lei , Xie Ding Bao. Formal Verification of Hybrid System: *Journal of Software*,2014,25(2):219−233
5. The Genesis and Development of Model Checking: *Fact vs. Fiction*
6. **自动机理论、语言和计算导引** ：【美】J E霍普克罗夫特 著
7. Symbolic model checking based on automatic theory.
8. Rüdiger Ebendt; Görschwin Fey; Rolf Drechsler (2005). *Advanced BDD optimization*. Springer. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-0-387-25453-1](https://en.wikipedia.org/wiki/Special:BookSources/978-0-387-25453-1).
9. Chieh-Jan Mike Liang, Börje F. Karlsson, Nicholas D Lane, Feng Zhao, Junbei Zhang, Zheyi Pan, Zhao Li, Yong Yu，SIFT: building an internet of safe things。 In Proceedings of IPSN (International Conference on Information Processing in Sensor Networks) 2015, ACM.
10. Croft, Jason, Ratul Mahajan, Matthew Caesar, and Madan Musuvathi. Systematically exploring the behavior of control programs. In Proceedings of the 2015 USENIX Conference on Usenix Annual Technical Conference, pp. 165-176. USENIX Association, 2015.
11. Bernd Becker; Rolf Drechsler (1998). *Binary Decision Diagrams: Theory and Implementation*. Springer. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [978-1-4419-5047-5](https://en.wikipedia.org/wiki/Special:BookSources/978-1-4419-5047-5).