2023 年 CCF —— 华为胡杨林基金
形式化专项
申报方向与课题

CCF-华为胡杨林基金技术委员会——形式化领域组

二零二三 年 四 月

第一条 课题分类

本专题中课题主要分为两类：开放课题和产业课题。

开放课题不限定具体研究内容，主要资助具有前瞻性、前沿性、能为产业全面升级储备能力，实现关键基础技术底座自主、领先的相关课题。

产业课题主要针对产业典型问题，持续提升形式化技术的能力上界、降低业界应用相关技术的门槛和成本，创造产业价值，形成本基金的正循环。

第二条 基金池

2023 年度形式化专题基金池共计 300 万人民币。

第三条 开放课题

单课题原则上资助额度不超过 15 万，为期一年。

鼓励开展同产业痛点、共性问题结合紧密的相关研究。鼓励进行前沿基础性的新研究方向和技术路径的探索。

第四条 产业课题

单课题原则上资助额度不超过 50 万人民币，为期一年。

提交成果中原则上需要包含源代码。

本年度拟资助的 5 个产业课题方向：

1、 自适应精度的抽象解释技术

2、 TLS 协议栈代码功能形式化验证

3、 基于局部搜索的程序轻量形式化验证技术

4、 分布式异步时序问题分析检测技术

5、 基于 RISC-V 实现的带可信执行环境处理器的形式化验证技术

自适应精度的抽象解释技术

课题背景

抽象解释是形式化验证中的一种常用手段。相对定理证明手段而言，抽象解释技术将程序语义在抽象域上进行重新解释，因此能实现对程序性质的更自动化的判定。

应用抽象解释的核心问题是抽象域的选择。不同的抽象域对应着对程序实际执行不同程度的近似程度：较粗糙的抽象域（如区间）有更高的执行效率，但会带来更多验证中的误报；较精细的抽象域（如多面体）虽然可以降低验证中的误报，但性能相对较差，较难应用到复杂程序中。

一些现有的验证工具（如 Astrée，Frama-C/EVA 等）支持多种不同精度的抽象域选择（如区间、八边形、多面体等），但其选择是通过在启动时设置参数实现的。这样既带来了抽象域的选择成本（需要专家进行实验、评估并判断），又不能自动化地在不同函数采用不同精度的抽象域。

因此，我们希望能够探索一种自适应精度的抽象解释技术，根据验证对象、验证目标，自动调整抽象解释的精度和结构，从而实现高效低误报的软件自动验证。

项目价值：

实现高自动化、低误报、高效率的抽象解释分析，扩展当前基于抽象解释技术的验证能力，更好地提升 C/C++代码的自动验证效率，降低验证工具的使用门槛；

研究内容

在抽象解释框架中，对程序语义的抽象化和具体使用的抽象域紧密相关。常用的抽象域往往经过精心设计，使其能在保存需要信息的同时实现尽可能高效的运算（如 join、widening 等等）。

Frama-C/EVA 中提供了按函数指定不同的抽象域的功能。为了实现抽象域的***自动选择***和***自适应调整***，需要研究以下问题：

- 如何对局部代码的自动评估，以判断何种抽象域更适合使用；
- 如何在验证出现报警时进行自动精化（选择更精细的抽象域），以排除可能的误报；

课题提供

- 典型的公司内抽象解释器使用场景；
- 与业务场景中代码功能相似的开源代码；
- 用于评估工具能力的 benchmark；

项目目标

- （必要）自适应精度的抽象解释技术的设计方案以及技术报告或论文，构建基于待验代码特性的抽象域自动评估能力和抽象域自适应精化能力；
- （必要）与 Frama-C 框架整合的内存安全性原型验证工具，在给定的 benchmark 上相比原有的 EVA 插件在自动化程度、稳定性、完备性等方面有更好的表现；
- （优选）具备扩展能力，支持用户自定义新的抽象域；
- （优选）在自动精化时支持增量验证，以提高精化过程的执行效率；

TLS 协议栈的代码功能形式化验证

课题背景

TLS 协议是网络认证和隐私的基础，其规范由 RFC 相关标准文档定义。基于这些自然语言的标准文档，不同的加密套件（如 OpenSSL、BoringSSL 等）提供 TLS 协议栈的具体实现，这些安全套件会被整合到系统中，为系统提供相关功能。为了保障实际系统的安全性，一个自然的问题是如何证明具体的代码实现的功能和 RFC 标准文档的规范一致。

我们希望能使用形式化验证的手段为目标加密套件的 TLS 实现提供严格保证。然而，现有的密码学相关的协议验证工作（如 Tamarin, ProVerif 等）往往基于较高层次的 Dolev-Yao 模型，更关注协议本身的高层次性质。为了进行 TLS 协议栈的代码级别功能验证，需要开发更底层的验证手段，能够直接进行面向代码的协议实现进行功能正确性验证。

项目价值：

- 为 TLS 协议的典型开源实现（如 OpenSSL 或 BoringSSL）的功能正确性提供严格形式化保证；
- 构建面向加密协议的、代码级别协议实现的功能验证能力；

研究内容

- TLS 协议标准的形式化建模：根据 RFC 标准文档构建 TLS 协议栈规范的形式化描述（参考模型）（如 AWS s2n 的验证工作[1]中提供的规约）；
- TLS 协议实现代码的形式化建模：提供源代码的形式化建模工具，据此构建实现代码的形式化模型描述（如 RefinedC 工具中提供的 C 代码到 Coq 代码的自动翻译能力）；
- 标准文档和实现代码的一致性证明：形式化证明目标 TLS 协议的代码实现的功能和相关 RFC 标准文档一致。

课题提供

- 需验证协议的相关标准文档；
- 待验证的 TLS 协议的实现；

年度目标

- 连通已有的相关形式化工具，打通流程并构建完整的代码级别功能验证解决方案，输出技术方案文档或论文；
- 对 TLS 协议栈中的关键协议（如 ECDH、ECDHE 等）提供形式化规约描述和配套工具（优选：形式化描述可执行，从而能够和其他实现进行交叉测试和验证）；
- 对目标加密套件实现的 TLS v1.3 整个握手流程进行形式化验证，保证其和标准文档的一致性（优选：验证自动化、连续化，可自动应对代码实现的小变更），给出验证方案设计文档、验证代码和验证报告；

[1] Andrey Chudnov, Nathan Collins, Byron Cook, Joey Dodds, Brian Huffman, Colm MacCárthaigh, Stephen Magill, Eric Mertens, Eric Mullen, Serdar Tasiran, Aaron Tomb, Eddy Westbrook: **Continuous Formal Verification of Amazon s2n**. CAV (2) 2018: 430-446

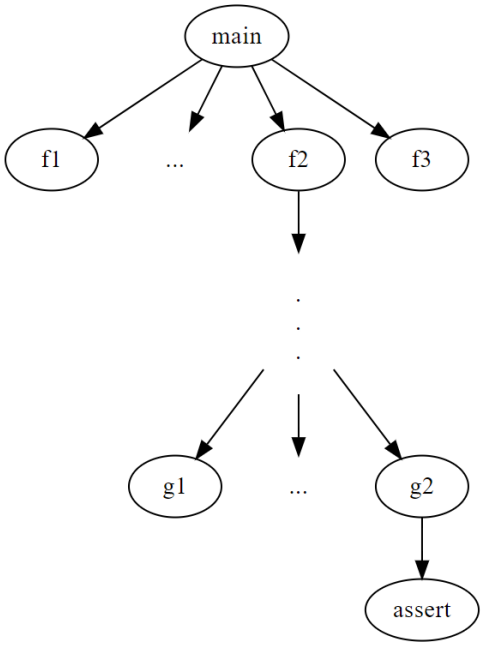
基于局部搜索的程序轻量形式化验证技术

课题背景

采用形式化方法验证软件代码的正确性是保障软件可靠性的有效途径。然而，静态形式化验证在实际工业界场景应用中面临的一个重要挑战是如何将误报降到最低，例如业界工具 Trust-In-Soft Analyzer(TIS)在对代码进行基于抽象解释的自动验证时需要使用者来判断/消除误报。误报分析高度依赖专业知识，但随着软件规模的增大，程序中的代码变得越来越复杂，手动进行误报分析给验证人员带来了巨大负担。因此，迫切需要探索一种高效的自动化技术方案，利用局部搜索等技术实现一种启发式的高效断言判定技术，辅助软件程序的形式化验证，提高自动化程度和效率，降低验证成本。

研究内容

在实际验证工作（如针对 C/C++代码的基于 TIS/Frama-C 的内存安全自动验证）中，确认未定义行为（UB）是一个非常重要的步骤。高效的反例生成可以快速确定 UB 是真实的还是误报。此外，当验证困难时，该技术还可以确定验证任务的不可行性，提高验证算法的效率。我们希望能够通过使用局部搜索等技术，实现一种启发式的高效断言判定技术，从而实现高效的形式化验证和反例发现。



具体而言，在验证过程中，证明代码中的断言是一个重要的步骤。当证明器无法证明一个断言时，通常会将其作为一个潜在问题报告，等待进一步确认。本项目希望申请人能够借助局部搜索等技术，针对大规模代码中的断言高效生成反例。

解决的关键问题：

使用约束求解和局部搜索技术，针对 C/C++ 代码中任意位置的 ACSL（https://frama-c.com/html/acsl.html）描述的断言，实现一种启发式的高效断言判定技术。

希望使用的技术手段和解决的问题：

我们希望采用区间算术、局部搜索、符号执行、约束求解和黑箱优化等技术来解决关键问题，例如复杂类型（如：字符串、数组、结构体指针等）数据结构符号化、断言路径查找、断言反例生成和处理黑箱函数。通过这些技术，我们期望实现一种启发式的高效断言判定方法，从而最终提高软件验证过程的整体效率和自动化程度。

课题提供

- 业务场景中典型（脱敏）代码；
- 与业务场景中代码功能相似的开源代码；

项目目标

- 复杂类型（如：字符串、数组、结构体指针等）数据结构符号化方法或技术；
- 有黑箱函数的断言路径查找和断言反例生成方法或技术；
- 高效反例发现技术设计方案及技术报告或论文；

- 可用的原型工具，在 10+万 C/C++ 代码规模下，对其中传统自动化形式化验证工具难以确认的 UB 进行有效的反例发现。

分布式异步时序问题分析检测技术

课题背景

分布式系统作为现代应用程序的基石被广泛部署，但它们的正确性、安全性和可靠性仍然很难保证。一个关键原因是，这些系统不仅需要执行复杂的存储和计算操作，而且存在高度的并发与异步，消息处理有前后时序的依赖关系。在分布式系统的所有问题类型中，复杂的消息通信交错导致的问题最难解决，并且因为网络的不可靠等因素而加剧，这些跨越多个节点的非确定性事件序列会导致“分布式异步时序问题”。这些问题很难被发现和诊断，并且常常导致数据丢失/不一致、业务失效等严重后果。此外，分布式异步时序问题的检测通常需要定义业务中的消息处理时序依赖模型，而现实中往往缺少这些时序依赖模型。因此，探索面向分布式异步时序问题的自动化分析检测技术具有非常重要的实际意义。

研究内容

研究和开发针对分布式异步时序问题的分析检测技术，其分析结果可以帮助开发者在编码阶段尽早发现和预防此类错误，或者在软件测试阶段为并发专项测试提供指导信息，以更好地提高分布式系统的正确性、安全性和可靠性。

- 针对 C/C++分布式系统中并发导致的异步时序问题，提供面向特定（一种或多种）性质的分析算法/工具。特定性质包括：数据发送/到达顺序异常、数据处理违反时序依赖、共享资源竞争等。
- 方案包括但不限于静态分析、动态分析、动静态结合等技术方法，需具备在工业级大规模分布式系统落地应用的可行性。
- 算法/工具兼顾通用性及可扩展性，支持业界标准的分布式并发同步源语（如 RPC、socket、消息队列等），并可通过扩展/适配的方式支持华为自研的分布式并发同步机制。

课题提供

- 基于产品代码的主要问题特征，抽象出来的测试 Demo。
- 业务场景中的分布式并发程序（现场调试）。

项目目标

以下为主要的研究目标，立项时可根据合作老师的建议并结合实际情况做出适当调整。

- 对于研究内容中提到的至少一种特定性质，构建动、静态分析技术有效快速地进行分析，准确率>80%。
- 能够满足分析工业级软件的性能要求：（1）静态分析能至少在五小时内分析一百万行代码；（2）动态分析不能引入超过十倍的运行时间开销，内存峰值开销小于 16G。
- 可以检测出测试 Demo 中符合至少一种特定性质的所有问题；在现场调试环节，能够发现产品代码中的实际问题。

开发和交付面向分布式异步时序问题的分析算法设计文档和工具（含代码）。

基于 RISC-V 实现的带可信执行环境处理器的形式化验证技术

课题背景

国内外对设备/芯片安全准入与法规要求日趋严格，行业的迅猛发展对安全要求逐步提升，如汽车行业对车载芯片具有高安全要求，对芯片开发工作提出了更高的挑战。基于开源指令集 RISC-V 架构设计处理器芯片，在底层设计时加入对可信执行环境（Trusted Execution Environment，TEE）的支持，能够有效提供高安全保障。然而，如何确保芯片完成了设计的安全预期，实现了设计功能同样至关重要。形式化验证技术可以应用在芯片需求、设计、实现等各开发阶段，有效保证各阶段输出的正确性，同时也是芯片产品通过国际 CC EAL 5+认证的必要技术。本课题的研究对象为基于 RISC-V 设计的带可信执行环境处理器，研究针对具体硬件语言 Chisel 的形式化建模验证技术。一般地，可通过将 Chisel 转 Verilog 代码后进行验证，但转化后的语言存在可读性差、状态空间变大等问题，为后续形式化验证带来更大困难。此外，状态空间爆炸问题导致形式化方法难以在实际大规模代码上应用，需要进行有效的模型抽象与验证技术。

研究内容

本课题拟研究面向基于 RISC-V 指令集架构设计的带可信执行环境处理器的形式化验证技术。

针对此问题拟开展包括但不限于如下研究：

- 从硬件实现语言 Chisel 自动转换成形式化模型，以支持模型检验、定理证明等形式化验证技术；
- 特定功能、安全性质提取与形式化描述，如可信执行环境（TEE）中内存隔离、安全读写访问权限等安全性质；
- 通过对性质分析，对转换后的形式化模型进行有效的抽象，缩小状态规模，支撑在处理器级别有效完成性质的形式化验证。

课题提供

- 产品硬件代码的主要特征；
- 可转换的开源 Chisel 代码，如 boom、rocket、香山处理器代码等。

项目目标

以下为主要的研究目标，立项时可根据合作老师的建议并结合实际情况做出适当调整。

- 针对芯片处理器实现的 Chisel 代码进行自动转换生成形式化模型，提供相应的工具及源码；
- 支持转换的 Chisel 程序规模达到数十万行，具备快速转化的能力，转换速度要超过通过转成 Verilog 代码，再转成形式化代码的途径；
- 对于研究内容中的特定功能、安全性质进行提取与形式化表达，能够在转换后的形式化模型上有效进行验证，对于违反性质的场景要提供反例路径。