2022 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——形式化专项组

二零二一 年 十二 月 十三 日

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。产业课题主要为了持续提升当前相关 State of the art 技术的能力上界，针对典型问题持续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

2022 年度形式化专题基金池共计 360 万人民币，其中开放课题 100 万人民币，产业课题 260 万人民币。

第三条 开放课题

单课题原则上资助不超过 25 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为 40-60 万人民币，为期一年。提交成果中原则上需要包含源代码。

2022 年度拟资助 5 个课题：

1、 C 程序循环不变式自动生成技术研究

2、 面向协议的自动/辅助建模技术研究

3、 具有可扩展性的并发程序静态分析技术研究

4、 具有可扩展性的代码路径可达性求解验证

5、 面向 Rust 程序和库函数的分析与测试技术研究

课题一：C 程序循环不变式自动生成技术研究

课题背景

采用形式化方法证明软件的正确性，是保障软件可靠性的有效方法。将形式化方法应用于软件代码验证时，核心的问题之一是处理循环。而要很好地理解、证明和推导循环程序，其重点是要能正确地写出循环程序的循环不变式，这通常需要人工参与。随着软件规模越来越大，程序中包含循环的代码往往多种多样，手动提供循环不变式给验证人员带来了较大的负担。因此，亟需探索具有一定自动/半自动化程度的技术方案，从循环程序的代码中智能推导和生成循环不变式，以辅助软件程序的形式化验证，提高自动化程度和效率，降低验证的成本。

研究内容

业务场景中的循环结构常常包含分支结构和嵌套循环，操作的数据类型除标量型变量外，还包含字符串、数组等向量型变量和链表，这类循环程序往往通过对被操作的数据结构的元素进行依次遍历来完成预期功能。

针对业务场景中 C/C++程序的循环结构和操作的数据类型，研究一种新的循环不变式的自动生成技术，以辅助实际应用程序的内存安全验证和功能性验证。

1. 需面向业务场景程序展开研究，以适应业务场景的需求。
2. 技术手段包括但不限于综合利用代码分析、抽象解释、约束求解和机器学习等技术。
3. 实现面向功能性验证和内存安全验证的循环不变式自动生成。
4. 孵化并开发循环不变式自动化生成技术或工具。

课题提供

- 部分样例程序（含 C 语言源代码、正确的循环不变式，以及待验证的性质）。
- 业务场景中包含循环结构的典型（脱敏）代码，或与业务场景中代码功能相似的开源代码。
- 需要循环不变式的验证工具（例如 CBMC 和 Frama-C）的部署方式和使用方式。

年度目标

以下为主要的研究目标，可根据合作老师的建议并结合实际情况做出部分调整。

- 对于操作字符串、数组、链表等数据结构的C程序，能为这类（至少同时处理上述数据结构中的两种）程序的循环代码自动生成正确的循环不变式。
- 课题提供的用例在语法能处理的情况下，生成的循环不变式可有效应用于特定性质的证明（e.g.，50%以上的循环不变式可用于特定性质的证明）。
- 交付技术方案和算法设计文档，以及为循环代码自动生成循环不变式的工具原型（含代码）。

课题二：面向协议的自动/辅助建模技术研究

课题背景

各类网络和通信协议已广泛应用于当今数字经济中，而且很多应用场景都是安全攸关的，如何通过形式化验证技术来确保这些协议设计的安全和可信也变得越来越重要。这些协议大都是以文本等方式描述的，如 3GPP 的 5G 协议等，目前对协议的形式化验证通常需要形式化专家对协议描述进行人工建模，然后将构造的形式化模型交给相应的工具进行形式化验证，导致对协议进行形式化验证的门槛和成本高，无法大规模应用。例如协议验证工具 Tamarin 和 ProVerif 在对协议的一些安全性质进行验证前都需要先由安全专家根据协议描述构建相应的形式化模型，并在此之上对待验安全性质构造形式化证明。在这类场景下，如何帮助工程师从协议描述自动化提取形式化模型具有非常重要的实用价值。

研究内容

本课题拟研究面向协议（文本）描述的自动或辅助建模技术。针对此问题拟开展包括但不限于如下研究：

- 协议的领域建模中间语言研究：如适配主流协议验证工具(如 Tamarin Prover、ProVerif 等)的面向普通工程师的领域建模中间语言设计与开发。
- 构建领域建模中间语言协议描述与协议的形式化模型之间的自动转换。
- 研究基于 AI/NLP 等技术的协议文本的模型提取技术：如从协议文本到上述领域建模中间语言协议描述或其他结构化协议描述的自动转换。

课题提供

- 主流协议验证工具的使用方式和部署方式。
- 协议建模与验证实际案例的相关学术文献。

年度目标

以下为主要的研究目标，本年度要求至少完成其中一项，立项时可根据合作老师的建议并结合实际情况对研究方向做出适当调整。

- 设计并开发一种可适配主流协议验证工具（如 Tamarin Prover）的协议建模领域语言。

一方面，该语言易用性强（可供普通工程师对协议进行建模）；另一方面，该语言也能够比较容易被自动转换为形式化模型。提供语言设计文档和工具环境。

- 构建上述协议建模领域语言协议描述到协议形式化模型的自动转换，并选取某类协议作为研究对象进行实验。
- 研究并构建基于 AI/NLP 等技术的协议文本的模型提取技术，并选取某类协议作为研究对象进行实验。

课题三：具有可扩展性的并发程序静态分析技术研究

课题背景

由于并发编程的复杂性和并发程序执行时因为线程交错引起的不确定性，编程人员在编程和调试阶段需要耗费大量的时间和精力，且较容易引入严重的并发错误（数据竞争、死锁、内存破坏等）。触发这些并发错误产生的条件往往都比较隐蔽和苛刻，不仅依赖特定的输入，而且还往往取决于特定的线程执行交错。因此难以被检测和修复。并发程序的分析受制于语义复杂、状态空间爆炸等问题，其分析效率和准确性仍具有较大的提升空间，尤其是对于复杂的大规模应用程序。因此，探索高效和具有可扩展性的面向并发程序的自动化分析技术（例如 Meta 的 Infer/RacerD 可以大规模快速地执行对于数据竞争的分析）就具有非常重要的实际意义。

研究内容

研究和开发具有可扩展性的并发程序静态分析技术，其分析结果可以帮助开发者在编码阶段尽早发现和预防并发错误，或者在软件测试阶段为定向测试提供指导信息，以更好地提高并发程序的正确性、安全性和可靠性。

- 针对 C/C++多线程程序，提供面向特定（一种或多种）性质的静态分析。例如检查数据竞争、识别潜在的竞争条件和可能发生错误的关键点、验证程序中不存在死锁、并发程序的路径可达性判定等。
- 实际应用程序中往往一个功能模块的代码就超过 10 万行，甚至上百万行，需面对调度策略组合爆炸、线程间别名分析和执行路径重现等方面的挑战，研究可扩展的静态分析技术。
- 孵化并开发并发程序静态分析框架或工具。

课题提供

- 产品代码的主要特征。
- 少量业务场景中的并发程序，或类似的开源并发程序。

年度目标

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一至两种性质，构建的静态分析技术能有效和快速地进行分析，在准确率上相比现有工具有较大提升。
- 支持的程序规模到十万行或百万行，有较好的可扩展性。
- 帮助分析产品代码或开源软件，并在其中发现实际的代码问题。
- 开发和交付面向并发程序的静态分析算法设计文档和工具（含代码）。

课题四：具有可扩展性的代码路径可达性求解验证

课题背景

代码路径可达性分析探究计算机程序状态之间的可达性关系，基于程序源代码或给定状态迁移规则,确定一个状态集合是否可以从系统中特定的初始状态集合到达。代码路径可达性分析广泛应用于静态缺陷检测、编译、代码建模、定向测试和软件安全属性的形式化验证等方面。

研究内容

目标：给定源代码，分析从初始状态到含一组特定代码位置的目标状态是否可达，并在可达情况下给出具体的执行路径和准确的触发输入。

过程：通过利用符号执行、约束求解、最弱前置条件推理、双向分析、搜索优化和程序归纳等技术手段，自动化分析从初始状态到特定代码位置的路径可达性。

方向一：研究一种适用于实际应用场景的分析技术，解决分析的准确性和可扩展性的问题。需面向大规模代码（千万条路径），快速且精准地判定路径可达性。

方向二：代码可达路径既定性质（比如：SQL注入、文件访问提权等）的自动验证是当前在大规模代码场景下形式化验证可行的研究方向之一，在一定程度上规避约束求解规模问题，研究成果可快速在实际产业环境下应用。

方向三：代码路径可达性分析问题最通常需借助SMT求解器进行约束求解，但实际场景下许多约束公式都无法在有限的时间内给出解答。研究专用的SMT求解器，集成并高效地组织各类理论求解方法，提升面向程序分析的 SMT 求解技术。

技术、算法不限。

课题提供

- 基于程序控制流程图的路径分析技术或工具，可分析出可能的程序路径（通常为千万条路径，包括可达和不可达的路径）。
- 少量业务场景中的脱敏代码，或与业务场景中代码功能相似的开源代码。

年度目标

通过新算法和技术方法解决软件代码模型节点间路径可达性验证、可达路径对应输入用例生成、可达路径既定性质的自动验证问题，提升软件关键路径的健壮性和安全性，开发并交付相关路径可达性分析工具，并且满足如下要求：

- 分析结果提供三种选项：可达、不可达、无法判定，其中无法判定的比例较低（e.g.，小于30%）。
- 可达路径具备对应数据集生成能力。
- 可达路径完成至少提供两种既定性质的自动验证（e.g.， SQL注入、文件访问提权等）。
- 理论创新方面，最好能在相关领域发表学术论文。

课题五：面向 Rust 程序和库函数的分析与测试技术研究

课题背景

Rust 是一门新型系统级编程语言，它以保证内存安全、并发安全和高性能而著称。Rust 正在传统的系统软件中得到越来越多的应用，如操作系统、嵌入式系统、网络框架、浏览器等，在这些领域，安全和性能都是不可或缺的。Rust 的安全规则是非常严格的，编译器通过类型检查和借用检查可以确保了 Rust（safe）程序的安全性。然而，在某些需要调用底层硬件系统，或需要获得更好性能的场景下，编程人员往往会使用不受编译器检查的 unsafe 代码。Unsafe 代码意味着，编译器的安全检查职责被暂时委托给了编程人员。Unsafe 代码的内存安全性对于整个程序的内存安全是至关重要的，因为大多数系统软件，如操作系统或标准库，都离不开它。当前，Rust 语言正在变得越来越流行，包含 unsafe 代码的 Rust 的应用程序和库也逐渐变多。因此，设计和开发实用的面向 Rust 的应用程序和库的分析与测试技术有助于提高整个 Rust 生态系统的健壮性和安全性。

研究内容

Rust 程序和库函数中存在不少 unsafe 代码，这些 unsafe 代码破坏了整个 Rust 程序的安全性。因此，彻底地分析与测试 Rust 程序和库函数以排除错误是至关重要的。尽管目前已经有了很多研究型项目，比如形式化类型系统和操作语义，建立模型用于验证程序的正确性。但是它们的可扩展性还不够强，难以覆盖到整个 Rust 生态系统。当前对于库函数的测试依赖于程序员手动构造测试用例，这是一个低效的过程，而 Rust 编译器严格的安全规则检查，也给自动化构造能通过编译的测试用例带来了一定的挑战。另一方面，Rust 编译器确保的安全性范围有限，允许一些可疑操作的检查，例如 Rust 编译器不能保证程序中没有死锁、内存泄露、整数溢出、并发时序问题等错误。因此提供多种技术与工具来检查和预防这些问题也是非常有必要的。根据以上挑战，研究方向的总结如下：

方向一：加强针对 unsafe 代码的内存安全检查，设计与开发可扩展的分析与验证技术。

方向二：研究面向库 API 的测试用例自动生成技术，可以用于标准库和主流库函数的自动化测试。主要的挑战是生成有效的 API 调用的序列，以及合成类型良好的 Rust 程序，以考虑 Rust 的所有权类型系统和多态性。

方向三：孵化并开发检查 Rust 程序是否存在死锁、内存泄露、整数溢出、并发时序问题等

错误的技术与工具。

课题提供

- 业务场景中广泛使用的开源 Rust 应用程序、库、以及工具链。
- 主流 Rust 程序分析工具的使用方式和部署方式。

年度目标

开发新算法和技术方法，可用于Rust程序和库函数的分析与测试，提升Rust生态系统的健壮性和安全性：

- 针对一种或多种类型的Rust程序错误，开发新的分析与测试技术，能比现有的主流技术和工具发现更多的程序错误，或更快地发现程序错误。
- 为Rust开源社区的应用程序和库发现实际的错误或漏洞。
- 提交算法设计文档，最好能在相关领域发表学术论文。
- 开发并交付面向 Rust 程序和库函数的分析与测试工具。