软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(11):4989−5007 [doi: 10.13328/j.cnki.jos.006684]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

共识协议的形式化验证研究现状与展望*

葛 宁[1,3], 贺俞凯[1], 翟树茂[1], 李晓洲[1], 张 莉[1,2,3]

[1](北京航空航天大学 软件学院, 北京 100191)

[2](北京航空航天大学 计算机学院, 北京 100191)

[3](软件开发环境国家重点实验室 (北京航空航天大学), 北京 100191)

通信作者: 张莉, E-mail: lily@buaa.edu.cn

摘 要: 分布式系统在计算环境中发挥重要的作用, 其中的共识协议算法用于保证节点间行为的一致性. 共识协议的设计错误可能导致系统运行故障, 严重时可能对人员和环境造成灾难性的后果, 因此保证共识协议设计的正确性非常重要. 形式化验证能够严格证明设计模型中目标性质的正确性, 适合用于验证共识协议. 然而, 随着分布式系统的规模增大, 问题复杂度提升, 使得分布式共识协议的形式化验证更为困难. 采用什么方法对共识协议的设计进行形式化验证、如何提升验证规模, 是共识协议形式化验证的重要研究问题. 对目前采用形式化方法验证共识协议的研究工作进行调研, 总结其中提出的重要建模方法和关键验证技术, 并展望该领域未来有潜力的研究方向.

关键词: 共识协议; 形式化验证; 限界模型检测; 定理证明; 布尔表达式可满足性理论; 可满足性模理论

中图法分类号: TP311

中文引用格式: 葛宁, 贺俞凯, 翟树茂, 李晓洲, 张莉. 共识协议的形式化验证研究现状与展望. 软件学报, 2023, 34(11): 4989–5007. http://www.jos.org.cn/1000-9825/6684.htm

英文引用格式: Ge N, He YK, Zhai SM, Li XZ, Zhang L. Formal Verification of Consensus Protocols: Survey and Perspective. Ruan Jian Xue Bao/Journal of Software, 2023, 34(11): 4989–5007 (in Chinese). http://www.jos.org.cn/1000-9825/6684.htm

Formal Verification of Consensus Protocols: Survey and Perspective

GE Ning[1,3], HE Yu-Kai[1], ZHAI Shu-Mao[1], LI Xiao-Zhou[1], ZHANG Li[1,2,3]

[1](School of Software, Beihang University, Beijing 100191, China)

[2](School of Computer Science and Engineering, Beihang University, Beijing 100191, China)

[3](State Key Laboratory of Software Development Environment (Beihang University), Beijing 100191, China)

Abstract: Distributed systems play an important role in computing environments. Consensus protocols are employed to guarantee consistency among nodes. Design errors in the consensus protocols might cause failure in system operation and bring catastrophic consequences to humans and the environment. Therefore, it is important to prove the correctness of consensus protocols. Formal verification can strictly prove the correctness of target properties in designed models, which is suitable for verifying consensus protocols. However, the expanding scale of distributed systems results in more complicated issues and challenges to formal verification of consensus protocols. The method for formal verification of the consensus protocol design and increase in the verification scale are significant research issues in the formal verification of consensus protocols. This study investigates the current research on the employment of formal methods to verify consensus protocols, summarizes the key modeling methods and verification technologies, and proposes future research directions in this field.

Key words: consensus protocol; formal verification; bounded model checking; theorem proving; Boolean satisfiability problem (SAT); satisfiability module theory (SMT)

---

* 基金项目: 国家重点研发计划 (2018YFB1402700); 国家自然科学基金 (61902011); 北京航空航天大学软件开发环境国家重点实验室开放课题 (SKLSDE-2021ZX-01)
收稿时间: 2021-10-21; 修改时间: 2022-01-10, 2022-03-11; 采用时间: 2022-04-01; jos 在线出版时间: 2022-05-24
CNKI 网络首发时间: 2023-04-28

分布式系统是组件位于不同联网计算机上的系统, 该系统中的节点通过相互传递消息来进行通信并协调自身的行为[1]. 分布式系统必须是可靠的, 协议中的设计缺陷或系统外部攻击造成的系统行为错误可能带来严重的后果. 尽管分布式系统测试的成本比例往往很高, 仍然难以避免其由于设计缺陷或外部攻击的原因失效. 例如: 在 2015 年的同一天内, 纽约股票交易所突然暂停交易、华尔街日报网站下线、联合航空的全部航班停飞, 造成这次严重网络事故的原因是运行软件的分布式系统出现了故障[2]. 分布式系统的质量保证具有一定的难度, 主要原因在于分布式节点间存在大量、频繁的数据交互, 外界环境引入不确定性, 系统内部的行为复杂, 容易产生数据不一致问题[3]. 因此, 分布式系统和区块链系统普遍采用共识协议 (consensus protocol) 用以保障分布式节点间的一致性. 区块链系统中, 每个节点都可以被认为具有独立行为, 同样需要共识协议保证各节点上的数据和行为的一致性. 如何保证共识协议自身设计的正确是一个关键问题. 形式化验证是保证软件设计正确性的重要方法, 如何采用形式化建模和验证技术证明计算节点在自身故障或受到外部攻击的情况下仍能达成数据一致, 是当前分布式系统领域的热点研究问题.

我们经过初步调研发现, 在共识协议的形式化验证研究中, 很多工作提出了面向特定共识协议的建模方法和验证技术, 然而大部分方法在验证规模方面都难以达到实际应用的要求. 本文拟全面地调研相关工作, 总结目前在共识协议建模和验证的研究工作中所采用的方法, 分析方法的局限性, 在此基础上总结面临的挑战问题和未来的研究趋势. 为此, 本文对发表文献进行了调研, 重点回答了 4 个问题: (1) 在共识协议的形式化验证工作中, 哪类协议更受研究者关注? (2) 共识协议中, 哪些性质容易验证, 哪些性质难以验证? (3) 在使用不同模型检测技术对共识协议进行形式化验证的过程中, 哪种技术的验证能力更强? (4) 针对共识协议的形式化验证问题, 未来的研究趋势是什么?

本文第 1 节介绍了共识协议的分类、主要性质, 并概述了形式化验证技术, 明确了本文的调研目标, 提出了研究问题. 第 2 节详细介绍了调研结果, 对比了采用显式状态模型检测、符号模型检测、基于 SAT/SMT 的限界模型检测、定理证明等技术对共识协议进行验证的方法. 第 3 节总结分析了当前研究的进展和趋势, 提出未来可能突破的研究方向, 为后续的研究提供参考. 第 4 节总结全文.

1 相关技术背景与调研问题

本节介绍共识协议的相关概念, 重点分析并总结了分布式共识协议的重要性质, 概述了本文调研中涉及的形式化验证技术, 并提出了 4 个调研问题.

1.1 共识协议分类

在分布式系统和区块链系统中, 通常依据系统对故障组件的容错能力将共识协议分为崩溃容错协议 CFT (crash fault tolerant) 和拜占庭容错协议 BFT (Byzantine fault tolerant)[4]两类. 组件宕机和通信链路故障通常被称为良性故障 (benign faults), 篡改消息内容、发送/拒绝发送消息等行为通常被称为恶意故障 (malicious faults) 或拜占庭故障 (Byzantine faults). CFT 类协议保证系统在组件宕机或通信链路发生故障/断开等情况下仍能达成共识[5], 典型的 CFT 类算法有 Lamport 的 Paxos[6]及其扩展和改进的变体算法、RAFT (reliable, replicated, redundant, and fault-tolerant) 协议[7]等. BFT 类协议保证分布式系统在有恶意节点篡改消息、恶意发送消息的情况下仍能达成数据一致, 典型的 BFT 类协议有 Lamport 的 PBFT[8], 以及大多数经典区块链共识协议. CFT 类算法往往只能容忍良性故障, 而 BFT 类协议则可以容忍恶意故障. 本文将分别调研 CFT 和 BFT 两类协议的形式化建模和验证研究工作. 为此, 我们选取典型 CFT 类协议 RAFT 和 BFT 类协议 PBFT (practical Byzantine fault tolerance), 对其原理进行介绍.

1.1.1 RAFT 共识协议概述

RAFT 是一种管理复制日志的一致性算法, 其算法过程可分为两个主要阶段, 包括领导者选举阶段和日志复制阶段. 图 1 描述了领导者选举阶段的算法行为. 在该阶段中, 分布式系统的节点被划分为 3 种角色, 分别是跟随者节点 (follower)、候选者节点 (candidate) 和领导者节点 (leader). 当 RAFT 协议开始运行时, 所有的节点状态都默认设定为跟随者节点, 不同节点的计时器初始化的超时时间不同, 最先结束计时的跟随者节点改称为候选者节点, 并向集群内其他全部节点发送拉票信息, 拉票信息包含该节点的最新日志状态. 当集群中半数以上的节点投票

通过了该拉票信息, 则该候选者节点成为领导者节点, 并开始向各跟随者节点定时发送心跳信息. 当集群内的任意节点在一定时间内没有收到心跳信息, 则重新发起领导者选举过程.

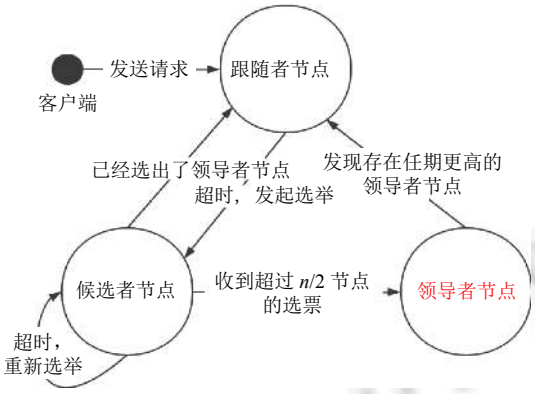


图 1 RAFT 协议中的领导者选举阶段

在日志复制阶段, 领导者节点向集群内其他节点发送添加日志消息, 收到该消息的节点会把该消息内部的日志添加到本地日志, 并向领导者节点返回 ack 确认消息; 当领导者节点收到全部节点反馈的确认消息后, 将发送应用日志消息; 收到该消息的节点将本地日志提交至本地状态机执行, 从而本地状态机执行命令后的状态与领导者节点执行命令后的状态相同, 达成分布式系统中各节点间的数据一致性.

为了保证协议安全性, RAFT 协议采用了维护最新日志编号的方法. 当集群内部某节点的最新日志编号与所维护的编号不一致时, 领导者节点将进行日志回滚, 并重新执行日志复制过程. 另外, 为了保证选举出的领导者节点具有最新的日志, 在拉票过程中, 节点只对日志比本地日志更新的候选者节点投票.

RAFT 通过领导者选举和日志复制使得分布式集群在无恶意节点进行攻击的环境下达成一致, 因此可以使用在区块链的私有链上. 然而如果有节点作恶, RAFT 协议无法达成一致. 这是 CFT 类协议的局限性. 在一些对容错分布式一致性算法的形式化验证中, 大多的验证工作倾向于关注良性故障或假设根本没有故障, 这些研究的对象便是 CFT 类算法. 但随着可容忍拜占庭故障的区块链的广泛使用, 对区块链共识协议的验证工作也逐渐兴起, 对 BFT 类协议的研究也在近些年逐渐增加.

1.1.2 PBFT 共识协议概述

PBFT 算法假设环境中存在恶意攻击. RAFT 算法的节点信任领导者节点, 出现的故障仅可能是良性故障; PBFT 算法则假设节点可能作恶, 即故意发送错误的消息等. PBFT 能容忍 1/3 的恶意节点.

如图 2 所示, PBFT 算法的主要过程可划分为 4 轮消息传递阶段[8], 分别是 pre-prepare 阶段、prepare 阶段、commit 阶段和 reply 阶段. Pre-prepare 阶段和 prepare 阶段是为了保证同一个主节点发送的请求在同一个视图 (view) 中的顺序的一致性, 其中视图在 PBFT 算法中起到逻辑时钟的作用: 当 PBFT 的主节点 (primary) 不能及时处理数据请求时, 视图也随之发生切换.

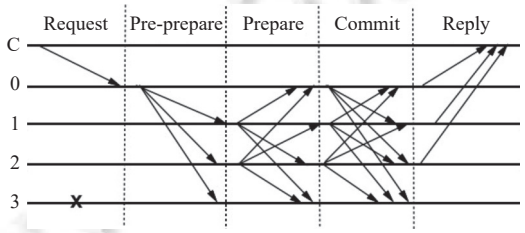


图 2 PBFT 协议执行流程

PBFT 的算法流程可以表述如下, 在一个节点总数为 n 的集群中, 假设最多有 f 个恶意节点. 在算法开始阶段, 主节点由 view mod n (即视图序号 view 取余集群规模 n) 选出, 在图 2 中以节点 0 表示. 在第 1 轮 pre-prepare 阶

段, 主节点将客户端 (在图 2 中以 C 表示) 发送的消息广播到集群内全部副本节点 (replica, 在图 2 中以节点 1、2、3 表示, 其中节点 3 模拟节点故障不发送消息的情况). 当其余副本节点收到消息后, 会对消息里包含的信息进行有效性检查, 例如检查该消息的数字签名、消息摘要是否正确, 消息编号是否在正确的范围, 以及视图编号是否和当前视图编号一致等, 如果消息通过检查, 则该副本节点进入 prepare 阶段. 在第 2 轮 prepare 阶段, 副本节点广播 prepare 消息并同时接收其他副本节点发送的 prepare 消息, 若收到 $2f$+1 个一致的 prepare 消息, 则该副本节点进入 commit 阶段. 在第 3 轮 commit 阶段, 副本节点广播并接收其他副本节点发送的 commit 消息, 若收到了 $2f$+1 个一致的 commit 消息后, 副本节点会执行消息中的请求, 并在最后 reply 阶段中把结果发给客户端.

考虑到主节点遭到恶意攻击而作恶的情况 (例如主节点遭到攻击后不再发送消息), PBFT 算法会提供视图切换 (view change) 技术来保证集群的可用性, 即: 副本节点每次收到请求时, 会启动一个计时器, 在正常收发消息时会不断重置该计时器, 但是在主节点宕机副本节点长时间收不到消息导致计时器超时的情况下, 该副本节点就会在当前视图中超时, 从而触发 view change 事件, 即该节点停止接受一切除 view change、new view change 等之外的请求, 同时广播 view change 消息请求将视图切换为 view+1. 当在 view+1 下计算得出的新主节点收到了 $2f$ 个有效的 view change 消息后, 广播 new view 消息, 将各个副本节点的视图切换为 view+1, 此时集群进入视图 view+1 状态, 主节点也发生切换.

1.2 共识协议的性质

在算法设计和程序运行中, 研究者会特别关注一些性质, 例如程序行为是否符合需求、是否存在异常、结果是否有效以及是否能在有限时间内完成计算等. 共识协议的性质可以包括安全性质 (safety) 和活性性质 (liveness) 两类[9]. 安全性是指系统不会进入非预期的状态, 即坏的事情不会发生; 活性是指系统运行时一定会达到预期状态, 即好的事情一定会发生.

一般地, 使用形式化方法验证分布式系统并对其做正确性证明工作的重点在于对运作在分布式系统上的一致性协议正确性的证明. 在验证层面上, 工作的重点往往在于验证算法的安全性和活性这两类性质. 根据 Lamport 的定义[9], 一个决定单值的共识算法满足如下性质会被称为具备安全性.

- 有效性: 只有被提出的值才有可能通过决议.
- 一致性: 最终只有一个值被选择并接受.
- 不可改性: 一个节点只有在决议达成后才可能知道最终选择的值.

活性性质则表达了共识协议关心的终止性, 即必须在有限的时间内达成一致.

在实际的异步网络中, 由于 FLP 不可能性 (Fischer, Lynch, Patterson 提出的分布式系统定理, 即在假设网络可靠、计算节点只会因崩溃而失效的最小化异步模型系统中, 仍然不存在一个可以解决一致性问题的确定性算法)[10], 异步网络中的一致性算法无法完全同时保证安全性和活性, 因此在实际工程中的算法会放宽对活性的要求. 例如 PBFT 算法设计就是在异步网络中不保证活性的情况下解决有恶意故障的共识问题[8].

使用形式化方法对共识协议进行验证, 能够严格证明共识协议的安全性. 针对活性, 往往通过条件限制放宽要求, 例如 Kwiatkowska 等人[11]的研究中采用了随机化的方法保证活性在一定概率下得以满足.

1.3 形式化验证技术简介

形式化验证是使用数学推理来验证需求与设计或实现是否一致的系统性过程[12]. 软件测试无法证明系统完全不存在缺陷, 也不能严格证明系统符合目标性质. 而形式化验证技术可以严格证明一个系统不存在某个缺陷或符合目标性质. 形式化验证技术主要分为定理证明、模型检测等类别.

定理证明[13]包括交互式定理证明 (interactive theorem proving) 和自动推理 (automated reasoning). 交互式定理证明通过用户与计算机相互协助完成证明[14], 采用的工具称为证明辅助工具 (proof assistant). 目前的证明辅助工具操作较为繁琐, 使用门槛较高[13]. 自动推理采用可满足性模理论 SMT (satisfiability module theory) 等技术自动地证明数学命题, 在推理和验证方面得到了广泛应用, 其局限性在于能够自动证明的数学命题较为有限, 对于复杂系统而言, 应用场景有限[13].

模型检测[15]技术将状态空间符号化表达, 并在符号化的空间上进行计算和遍历. 它将系统建模成有限状态系统, 系统的性质表达为计算树逻辑 CTL (computation tree logic)[16]、线性时间逻辑 LTL (linear temporal logic)[17]等时序逻辑公式, 而后在此模型上验证性质表达式的正确性. 模型检测相比交互式定理证明的优势在于可以自动执行性质验证, 无需人工干预. 模型检测技术可以划分为 4 类, 包括显式状态模型检测 (explicit-state model checking)、符号模型检测 (symbolic model checking)、基于可满足性理论的模型检测 (SAT-based model checking)、基于可满足性模理论的模型检测 (SMT-based model checking)[18]. 4 种方式各有优缺点, 但同时都有状态空间爆炸的问题.

显式状态模型检测通过列表或者表格的形式描述存储状态空间[18], 因此随着待检测系统规模增大, 需要遍历的模型状态数也会增加. 由于硬件性能所限可能难以遍历全部状态, 导致模型检测状态爆炸问题[19]. 减少状态空间的方法通常包括状态空削减[20]、存储空间压缩[21]、并行和分布式计算[22]、随机化和启发式搜索[23]等. 符号模型检测对显式模型检测的状态描述形式进行优化, 使用布尔函数来表示模型状态集合和状态转换关系, 并使用二叉决策图 (binary decision diagram, BDD) 在计算机中储存布尔函数, 从而大幅缓解了状态空间爆炸问题, 提升了模型检测的验证规模和验证能力. 可满足性理论, 也称 SAT 问题, 广泛应用于计算机科学、复杂性理论、密码系统、人工智能等领域. SAT 问题的基本形式指给定一个命题变量集合 X 和一个 X 上的合取范式 $\varphi(X)$, 判断是否存在一组真值赋值 $t(X)$, 使得 $\varphi(X)$ 为真. 如果找到 $t(X)$, 则称 $\varphi(X)$ 是可满足的 (satisfiable), 否则称 $\varphi(X)$ 是不可满足的 (unsatisfiable). 基于 SAT 的模型检测称为限界模型检测 (bounded model checking). 基于 SAT, 可以实现基于 k 阶推导 (k-induction) 的自动推理. 可满足性模理论 SMT 的求解对象是命题逻辑公式, 通过判定逻辑公式在组合背景理论下的可满足性来证明相关性质. SMT 的背景理论使其能很好地描述实际领域中的各种问题, 结合高效的可满足性判定算法, SMT 在测试用例自动生成、程序分析与验证、线性逻辑约束公式优化问题求解等研究领域中具有优势. 与 SAT 相比, SMT 引入了一阶逻辑, 因此 SMT 比 SAT 抽象程度更高, 表达能力更强. 基于 SMT 的限界模型检测技术同样存在状态爆炸的问题, 一般会使用数据抽象等技术缓解该问题.

在本文中, 为了方便表达, 我们将调研文献里用到的形式化验证技术总结为 3 类: 常规模型检测技术、基于 SAT/SMT 的限界模型检测技术以及定理证明技术. 其中常规模型检测技术包括显式状态模型检测技术和符号模型检测技术.

1.4 研究问题

为了深入理解共识协议形式化验证研究的现状和未来发展趋势, 我们提出了以下 4 个研究问题 RQ (research questions).

RQ1: 在共识协议的形式化验证工作中, 哪类协议更受研究者关注? CFT 和 BFT 类共识协议中都包含众多子类协议, 发表文献中只验证了部分协议. 有必要对这些工作进行调研, 分别总结 CFT 和 BFT 类共识协议验证工作的发展趋势和该研究方向的热度变化情况.

RQ2: 采用模型检测技术验证共识协议, 哪些性质容易验证, 哪些性质难以验证? 共识协议的性质包括属于安全性的一致性、有效性、不可改性及属于活性的终止性. 我们关心形式化验证技术更容易验证哪些性质、难以验证哪些性质, 以便在未来的研究中重点分析那些难以验证的性质的特点, 提出有针对性的验证方法. 本文考虑的 4 类性质是各类共识算法验证中普遍关心的性质, 并未覆盖某些共识协议中特有的性质.

RQ3: 基于不同的模型检测技术验证共识协议的一致性, 哪种技术的验证能力更强? 共识协议中最重要的性质是一致性, 我们调研的 80% 工作采用了模型检测技术进行一致性验证. 模型检测技术对于共识协议各类性质的验证能力如何? 现有的验证技术能够解决的问题规模是多大? 此外, 验证效果更优的工作采用了哪些关键方法和技术?

RQ4: 共识协议的形式化验证方向未来的研究趋势是什么? 通过本文的研究, 我们希望能够得出形式化验证共识协议领域的未来研究发展方向.

2 共识协议的形式化验证技术

本文对共识协议的形式化验证研究进行了调研, 表 1 总结了该方向的总体研究情况[2,11,24–47], 包括共识协议类

型、协议算法变体 (主要针对 Paxos 算法, 列出了被验证的 Paxos 的变体算法)、验证方法 (包括常规模型检测、基于 SAT/SMT 的限界模型检测、定理证明、数学证明). 本文中, 常规模型检测包含显式状态模型检测和符号模型检测, BMC 代指基于 SAT/SMT 的限界模型检测技术. 该表总体概括总结了部分共识协议的形式化验证技术. 在本节中, 我们将在每一小节介绍采用这些技术的形式化验证工作, 并对其进行总结和分析.

表 1　共识协议的形式化验证工作统计表

类别	共识协议	协议算法的变体	验证方法
CFT	Paxos	Paxos	常规模型检测[34]
		LastVoting	定理证明[24] BMC[25,35−37]
		UniformVoting	定理证明[24], BMC[35], 常规模型检测[26]
		DLS	定理证明[24]
		OneThirdRule	定理证明[24], BMC[35,36], 常规模型检测[26]
		Hybrid-1(α)	BMC[25,36,37]
		Multi-Paxos	定理证明[43]
		CordUniformVoting	BMC[25,35]
		Ceph的共识算法	常规模型检测[29]
—	FloodMin	—	BMC[35]
	Aspnes&Herlihy's algo		常规模型检测+定理证明[11]
	FTDA benchmarks: FRB、CBC、NBAC、NBACC		BMC[39]
	RAFT		定理证明[2]
	The Mostefaoui-Raynal algorithm		常规模型检测[27]
	The Chandra-Toueg Algorithm		常规模型检测[27]
	Taurus的共识协议		常规模型检测+定理证明[28]
BFT	FTDA benchmarks: STRB、ABA	—	BMC[34,39]
	FTDA benchmarks: BOSCO		BMC[34,40,47]
	FTDA benchmarks: C1CS、CF1S		BMC[34,47]
	A(T, E)		BMC[35], 定理证明[38]
	U(T, E)		BMC[35], 定理证明[38]
	Stellar consensus protocol (SCP) 区块链协议		常规模型检测[30]
	Echo multicast		常规模型检测[34]
	Algorand区块链协议		定理证明[41]
	Hybrid reliable broadcast		BMC[40]
	Byzantine fast Paxos		BMC[40]
	Tendermint		常规模型检测[32], 数学证明[46]
	CKB共识协议		常规模型检测[31]
	CKB块同步协议		定理证明[42]
	Snow White		数学证明[44]
	Ouroboros Praos	Ouroboros	数学证明[45]
	STBC	—	常规模型检测[33]

2.1　基于常规模型检测技术的共识协议验证

基于显式状态模型检测和符号模型检测进行共识协议验证的工作开展得较早, 大部分工作关注 CFT 类共识协议的形式化验证. 表 2 总结了基于显式状态模型检测和符号模型检测的研究, 包括算法类型、验证性质类型、采用的验证工具、提出的建模思想、相关文献、验证规模 (节点数)、在既定资源限制下的验证花费时间等. 进行形式化验证前首先需要对协议进行建模, 如何在建模中选取恰当的抽象方式, 既准确描述协议的行为, 又保证不引

入过多状态, 是许多研究工作中需要解决的重要问题. 我们按照不同的建模方式将表 2 中的研究工作划分为基于轮、基于 TLA+、基于时间自动机、基于 CSP 以及自定义语言建模方法. 本节将依据此分类对共识协议的验证工作进行讨论.

表 2 基于显式状态和符号模型检测的研究工作

类别	算法	验证性质	验证工具	建模思想/语言	文献	验证规模(节点数)	验证花费时间 (s)
CFT	Aspnes&Herlihy's algo	一致性 有效性 概率终止性	Candence SMV 定理证明	—	[11]	—	—
	The Mostefaoui-Raynal algorithm	一致性	SPIN	建模思想: 基于轮的思想	[27]	3	
	The Chandra-Toueg Algorithm					4	
	Paxos	一致性 有效性 终止性	MP-Basset (MP for message-passing)	建模思想: quorum transitions 语言: Java-like Message Passing	[34]	6	12 600
	LastVoting	一致性 终止性	NuSMV SPIN ALV	建模思想: Heard-Of模型	[25]	NuSMV: 4 SPIN: 3 ALV: 3 NuSMV: 4	NuSMV: 167 SPIN: 2 922 ALV: 1 921 NuSMV: 41
	OneThirdRule	一致性 终止性 有效性	TLC	语言: TLA+	[26]	4	1 546
	UniformVoting	一致性 终止性 有效性	TLC	语言: TLA+	[26]	4	1 330
	Taurus的共识协议	安全性等11个性质	TLC+TLAPS	语言: TLA+	[28]	DC模块: 4 CM模块: 6	DC模块: 805 CM模块: 913
	Ceph的共识协议	安全性	TLC	语言: TLA+	[29]	4	
BFT	Stellar consensus protocol区块链协议	一致性 有效性 终止性	UPPAAL	语言: 时间自动机 语言: C-like语言	[30]	5	—
	Echo multicast	一致性	MP-Basset (MP for message-passing)	建模思想: quorum transitions 语言: Java-like Message Passing	[34]	6	151 260
	Tendermint	一致性 有效性	PAT	语言: CSP	[32]	4	—
	CKB共识协议	一致性	UPPAAL	语言: 时间自动机	[31]	—	—
	STBC	安全性容错性领导者可信性验证者可信性	PAT	语言: CSP# LTL	[33]	9	—

2.1.1 基于轮的建模方法

基于轮的建模思想结合了基于消息传递的共识算法过程分阶段的特点, 按照每轮顺序执行的方式对协议行为进行建模. 这种抽象方式保留了协议的交互行为, 简化了消息传递的细节, 因此可以在一定程度上减少验证的状态空间.

Charron-Bost 等人[24]根据异步共识协议节点间通信闭合的特征, 在以往基于轮的建模方法的基础上提出了一种新的基于轮的建模思想, 结合 RRFD 模型[48]和传输故障模型[49]的优点, 提出了 Heard-of (HO) 模型, 并采用 HO

建模思想描述了 CFT 类的 Paxos 算法, 称为 LastVoting 算法. 尽管从计算角度来看, HO 模型相较于同类模型并没有明显优势, 但从容错分布式算法的建模和正确性证明的角度而言, HO 模型更加简单[24]. Tsuchiya 等人在后续的研究中沿用了 HO 建模思想, 使用 3 种不同的显式状态和符号检测模型检测工具验证 LastVoting 的一致性和终止性[25]. 针对一致性, NuSMV 的验证规模为 4 个节点, SPIN 和 ALV 的验证规模为 3 个节点; 针对终止性, 只有 NuSMV 可以完成验证, SPIN 和 ALV 很快出现了状态爆炸情况, 未能完成验证. Chaouch-Saad 等人还使用了 TLA+描述了 OneThirdRule 和 UniformVoting 的 HO 模型[26], 使用 TLC model checker 进行验证, 但验证能力同样仅达到 4 个节点. 在这两项工作中, HO 模型在建模层面将验证问题规模减小为验证算法单阶段的性质, 使未来的工作可以更加高效地使用限界模型检测技术.

Noguchi 等人[27]在后续研究中采用了另一种基于轮的思想对 CFT 类的 The Mostefaoui-Raynal Algorithm 和 The Chandra-Toueg Algorithm 算法进行建模, 并验证了其一致性. 在该工作中, Noguchi 等人[27]通过验证单一轮内的协议行为保证所有轮内行为的正确性, 有效缓解了状态空间爆炸问题. 该工作采用 SPIN 模型检测器, 2 个算法的验证规模分别达到 3 个节点和 4 个节点. 研究的结论表明, 基于显式状态和符号模型检测验证共识协议, 在集群规模不大的情况下就会遇到状态空间爆炸的问题, 其验证规模有限.

2.1.2 基于 TLA+的建模方法

Chaouch-Saad 等人[26]对 CFT 类的 OneThirdRule、UniformVoting 算法[25]进行了验证. 与 Charron-Bost 等人[24]采用 HO 方法的研究不同, 他们重点探索建模方案, 基于 TLA+以更高的抽象程度描述 HO 数学模型, 并使用 TLC model checker 进行验证. 针对状态爆炸问题, 该研究沿用了 Bernadette 等人提出的思想, 只对算法的一次粗粒度执行进行验证. 该工作中对一致性、有效性和终止性的验证规模为 4 个节点, 验证时间在 22–25 min 左右.

Gao 等人[28]则对云计算中的分布式数据库服务进行了研究, 并使用了形式化方法来对工业级分布式数据库 Taurus 进行了描述与验证. 该数据库使用了多个共识协议, 其中核心为多版本并发控制 (multi-version concurrency control) 以及基于 RAFT 的集群管理协议. 该工作使用了 TLA+对系统建模, 并与 Taurus 工程师沟通, 提取了包括安全性在内的共计 11 个性质, 用 TLC model checker 与定理证明系统 TLAPS 来验证这些性质. 在模型检测技术的验证中, Taurus 的 DC (distributed consensus) 模块验证规模为 4 个节点, 验证时间为 805 s; CM (cluster management) 模块验证规模为 6 个节点, 验证时间为 913 s. 在定理证明系统 TLAPS 的验证中则验证了 10 条性质. 这项工作成功地验证了 Taurus 的抽象模型, 并且向工业界证明了形式化方法在工业级系统上的有效性.

Fernandes[29]在其硕士学位论文中对最常用的开源分布式存储平台 Ceph 进行了研究. 共识算法是 Ceph 的核心, 该共识算法是与 Multi-Paxos 类似的 CFT 类算法, 因此 Fernandes 使用 TLA+对其进行形式化建模, 并使用 TLC model checker 对安全性进行了形式化验证. 其验证规模达到了 4 个节点. 在未来的工作中, Fernandes 希望能够验证活性, 另外还希望能够对 BFT 类共识算法进行建模验证.

2.1.3 基于时间自动机的建模方法

针对 BFT 类协议, Yoo 等人[30]对 SCP (Stellar consensus protocol) 区块链协议[50]的安全性和活性分别进行了形式化验证. SCP 使用区块链的 Stellar 加密货币算法, 用于联邦拜占庭协议. Yoo 等人使用时间自动机进行建模, 使用 UPPAAL 工具进行验证, 验证规模达到 5 个节点. Yoo 等人提供的抽象技术有助于处理 SCP 在形式化验证中的极大状态空间, 且该方法同样适用于对各种区块链平台的其他共识协议进行建模. 同时 Yoo 等人也指出 UPPAAL 对异步协议验证的可扩展性有限.

CKB (common knowledge base) 是 Nervos networks 的第 1 层, 它保证区块链的安全性, 包含一系列协议, CKB 共识协议是其中之一. CKB 共识协议是比特币共识算法的变体, 也是基于 PoW (proof-of-work) 的共识协议, 相比比特币共识算法有更高的事务处理吞吐量和更强的抵御自私挖矿攻击 (selfish mining attack) 的能力. Sun 等人[31]使用时间自动机对 CKB 协议进行建模, 并使用 UPPAAL 工具验证了 CKB 协议的正确性和一致性, 并通过模拟恶意攻击首次证明了 CKB 算法对自私挖矿攻击的抵御能力. Sun 等人计划在未来的研究中继续优化该技术, 以检测共识协议在其他恶意攻击下的鲁棒性.

2.1.4 基于 CSP 的建模方法

Thin 等人[32]对 BFT 类协议 Tendermint 算法进行了验证, Tendermint 是一种有拜占庭容错的 proof-of-stake 算法, 该研究使用了 CSP#和 PAT model checker 进行形式化建模与验证, 分别在 3 个节点和 4 个节点时验证了 Tendermint 算法是否会产生死锁, 算法达成共识的能力, 容忍无效块、审查攻击等错误的能力. 研究验证了 Tendermint 共识协议是无死锁的, 当至少 2/3 的网络达成协议时, 能够达成共识; 研究还证明了构成 1/3 以上的节点可以阻止网络达成共识, 并得出了算法在可用性方面存在一些不足的结论. 但该模型的验证规模较小, 仅能验证少量节点时算法的性质, 节点数量增加时会出现状态空间爆炸的问题.

Afzaal 等人[33]对基于区块链的众包 (crowdsourcing) 共识协议进行了形式化建模与验证. Afzaal 等人通过对现有的一些区块链共识协议的研究, 发现其普遍存在一定的局限性. 如 PoW 协议高度资源密集且难以应用在大型在线系统; 而在众包系统中应用 PoS (proof-of-stake) 协议则可能会损害节点的权利, 因为节点可能会受到安全性、公平性和中心化问题等的影响; 此外, 一些基于区块链的众包共识协议如 PoT (proof-of-trust) 协议也存在着各种局限. 因此, Afzaal 等人研究并提出了 STBC (secure and tustworthy blockchain-based crowdsouring) 共识协议, 并且使用 CSP#对该协议进行形式化建模, 使用 LTL 线性时序逻辑表达安全性和可信性等性质, 最后使用 PAT model checker 对该模型的安全性、容错性、领导者可信性、验证者可信性的性质进行形式化验证. 其中, 安全性的验证考虑了宕机错误和拜占庭错误. Afzaal 等人提出的 STBC 协议可以适用于任意数量的节点, 但是在节点数增加的同时验证时间也会随之增加并最终遇到状态爆炸问题. 在本工作中 PAT 的验证能力能达到至少 9 个节点, 但未给出上限. Afzaal 等人也表示计划在未来的工作中将会使用不同的技术来提高验证效率, 例如使用二叉决策图来进行符号模型检测, 还有偏序规约技术、基于反例的抽象精化技术, 以及限界模型检测技术.

2.1.5 基于自定义共识协议领域语言的建模方法

一些工作面向特定共识协议定义了领域建模语言, 并基于该语言建模并验证共识协议. 自定义建模语言的优势在于可以面向证明过程对协议进行抽象描述. Bokor 等人[34]对 CFT 类的 Paxos 算法以及 BFT 类的 echo multicast 算法进行了建模验证. 该研究使用类 Java 语言 MP (message-passing) 对协议进行了抽象建模, 在 Basset model checker 的基础上构造了基于偏序规约的 MP-basset model checker, 有效缓解了状态空间爆炸的问题. 该工作对 Paxos 的一致性、终止性和有效性进行了验证, 验证规模为 6 个节点, 共耗时 3 h. 对 echo multicast 的一致性进行了验证, 验证规模达到 6 个节点, 验证时间为 42 h. 该研究扩大了集群的验证规模, 并期望在符号模型检查领域研究中继续研究偏序规约技术.

在验证优化方面, 可以采用状态削减、组合方法等通用的模型检测状态空间压缩技术. 此外, 还需要研究面向特定共识算法的特定性质的验证优化技术. Kwiatkowska 等人[11]验证了 CFT 类的 Aspnes&Herlihy's algo 算法. 该工作认为, 在 FLP 不可能性的条件下, 终止性的要求无法达到, 因此选择保证一定概率的活性. 针对活性中的终止性条件, 他们提出了概率无等待终止性 (probabilistic wait-free termination). 该工作使用 Cadence SMV 工具对算法的一致性和有效性进行了验证; 对于概率无等待终止性, 则通过数学推理方法手工证明了协议在多项式时间内可终止. 在 2001 年, Kwiatkowska 等人[11]自动地验证了一个复杂的随机的分布式算法, 展示了如何对分布式算法进行自动化验证, 对后续的研究有较大的参考价值.

综合以上研究, 本文得出结论: 建模方法与验证能力的关系密切. 在验证技术中, 规模受限明显的是显式状态模型检测和符号模型检测技术. 这两种技术在验证大规模 CFT 类和 BFT 类共识协议时都容易发生状态爆炸问题, 导致无法验证或者验证失败, 因此普遍用于小规模集群的共识协议验证.

2.2 基于 SAT/SMT 限界模型检测技术的共识协议验证

模型检测中的很多问题都可以转化为 SAT 问题进行求解. 在 SAT 的基础上引入一阶逻辑, 则可满足性问题称为 SMT 问题. 近年来, 限界模型检测已经成为模型验证领域的研究热点[18]. 在本节中, 我们按照建模思想和验证技术对采用 SAT/SMT 技术的研究进行分类. 表 3 总结分析了目前基于 SAT/SMT 的共识协议验证研究情况, 包括算法类型、验证性质类型、采用的验证工具、提出的建模思想、相关文献、验证规模 (节点数)、验证花费时间等. 本文将这些验证方法归纳为 3 类, 分别为基于 HO 思想、基于 ByMC 工具、和基于 FOL (first-order logic) 的验证方法.

表 3 基于 SAT/SMT 限界模型检测的共识协议验证

共识算法	验证性质	验证工具	建模思想/语言	文献	验证规模 (节点数)	验证花费时间 (s)
FTDA benchmarks: FRB、CBC、NBAC、NBACC、STRB、ABA	不可改性 终止性	ByMC+SMT solver, NuSMV/SAT solver Plingeling	Promela扩展语言	[39]	—	—
FTDA benchmarks: FRB、CBC、NBAC、NBACC、STRB、ABA、C1CS、CF1S	一致性 有效性	ByMC, SMT solver Z3	Promela扩展语言	[47]	—	—
LastVoting	一致性	SMT solver Yices[25,36,37] SMT Solver Z3[35]	HO	[25,35,36,37]	8[36,37] 9[25]	10[37], 25[36], <0.1[35] (节点数未知)
	终止性				11[37], 14[25]	32[37]
Hybrid-1(α)	一致性				8[37], 7[36,25]	539[37], 945[36]
	终止性				11[37], 14[25]	121[37]
OneThirdRule	一致性			[35,36]	7[36]	7578[36]
CoordUniformVoting	一致性 终止性	SMT solver Yices[35,25]		[35,25]	13[25] 15[25]	<0.1[35] (节点数未知)
A(T, E)	一致性 终止性	SMT solver Z3	HO扩展	[35]	—	<0.1[35] (节点数未知)
U(T, E)	一致性 终止性					
FTDA benchmarks: BOSCO	一致性 有效性	ByMC (前端处理), SMT solver Z3 (后端模型检测)[47] Z3、CVC4 SMT model checker[40]	Promela扩展版[47] FOL模型[40]	[40,47]	—	—
Hybrid reliable broadcast	一致性 有效性 终止性	Z3、CVC4 SMT model checker	FOL模型, Ivy语言[40]	[40]	—	—
Byzantine fast Paxos	一致性 有效性	Z3、CVC4 SMT model checker	FOL模型, Ivy语言[40]	[40]	—	—

2.2.1 基于 HO 思想的研究工作

研究者们扩展了 HO 建模验证体系[25,27,35−37]. 在建模方面, HO 模型只需为每个节点和每一轮定义消息发送和状态转换函数, 即可简单地定义一个 CFT 类共识算法[38]. 在数学证明方面, 在 HO 模型早期的研究中, Schiper 最先采用数学推理的方式证明了共识协议的性质.

在自动证明方面, 在常规模型检测技术研究的基础上, Tsuchiya 等人[37]为了缓解状态爆炸问题, 采用基于 SMT 的限界模型检测来缩减状态空间, 利用 k-induction 自动定理证明进行验证, 将目标问题转化成了仅需验证算法执行一阶段的小规模验证问题, 并验证了 LastVoting、Hybrid-1(α) 等算法的一致性和终止性. 对于 LastVoting 算法, 其对一致性的验证能力达到了 8 个节点, 对终止性的验证能力达到了 11 个节点, 超过了基于常规模型检测技术验证的 4 个节点的验证规模, 说明该方法对验证规模的提升较为显著. 此外, Tsuchiya 等人还在后续研究中研制了 HO 语言的转换器, 自动地将 HO 模型转化为 Yices 求解器的 SMT 语言, 进而自动验证性质. Tsuchiya 等人计划在未来的研究中继续完善该技术, 并将使用此技术处理更多种类的共识算法[25].

大部分基于 HO 模型的研究主要聚焦 CFT 类算法的验证, 即仅考虑良性故障. 在作者的后续研究中, 还对 HO 模型做了扩展, 对 BFT 类算法进行建模, 并通过数学证明方法进行了验证[51].

HO 模型对形式化验证共识协议的研究产生了一定的影响, 有不少研究使用了 HO 模型描述的算法, 也有一

些研究对 HO 模型进行扩展. Drăgoi 等人在 HO 模型的基础上添加了一阶霍尔逻辑和半决策过程来完成共识算法性质证明, 对 HO 语境下的 BFT 类算法进行了验证, 一定程度上减少了验证时间[35].

2.2.2 基于 ByMC 工具的研究工作

由于分布式系统中存在拜占庭错误和崩溃错误, 一个节点不能等待来自特定节点的消息, 因此大部分共识协议通过计数器记录接收消息的数量, 并通过引入阈值条件 (threshold guard) 保护接收消息的过程, 例如接收到超过半数的消息时就做出某些行为, 这类共识算法被称为基于阈值保护的分布式共识算法 (threshold-guarded distributed algorithms). Konnov 等人针对基于阈值保护的分布式共识算法提出了参数化模型检测方法 PIA (parametric interval abstraction)[39,47,52−55], 将总节点个数和错误节点个数作为参数进行建模, 首次实现了在大规模分布式系统上验证共识算法的性质.

Konnov 等人为了实现自动化的参数模型检测, 引入了一系列技术以缓解模型检测的状态爆炸问题:

• PIA data (parametric interval data abstraction): 数据抽象技术, 对共识算法中的阈值进行抽象, 使用区间替换阈值条件, 最终得到了有限状态的系统.

• PIA counter (parametric interval counter abstraction): 计数器抽象技术, 对共识算法中的计数器取值进行抽象, 使用区间构成的抽象域替换整形计数器的值域, 将计数器取值限制在有界范围内, 得到了有界的状态空间.

• POR (partial order reduction): 偏序规约技术, 通过剪枝减小了求解器所需遍历的空间.

Konnov 等人[39]构造了集成建模和验证技术的 ByMC 工具, 并在工具中实现了上述技术. 在建模方面, ByMC 工具的前端接受拓展 Promela 作为输入, 能够支持参数化建模. 在验证方面, ByMC 提供以下 2 种工作模式.

• 在预处理阶段引入 PIA data 和 PIA counter 技术, 生成有限状态系统, 调用后端求解器进行验证. 若后端采用显式状态模型检测工具 SPIN, 或采用 NuSMV/NuXMV 中实现的基于 BDD 的符号模型检测算法, 则可以验证共识算法的安全性和活性. 若采用 NuSMV/NuXMV 中实现的基于 SAT 的有界模型检测算法, 则可以验证共识算法的安全性. Konnov 等人通过该种工作模式首次实现了分布式共识算法的参数化模型检测, 并成功地验证了数个不曾被自动化验证的分布式共识算法.

• 在预处理阶段引入 PIA data 和 POR 技术, 减小求解器所需遍历的空间, 并调用 SMT 求解器进行验证. Konnov 等人在案例验证中证明了基于 SMT 和偏序规约的有界模型检测技术 (SMT) 比基于 SAT 的有界模型检测 (NuSMV-SAT)、基于 BDD 的符号模型检测 (NuSMV-BDD)、基于显示状态模型检测 (SPIN) 表现更好, 并成功地验证了这些技术无法验证的 7 个分布式共识算法的安全性.

Konnov 等人后续的研究对 ByMC 工具进行了更多的扩展, 例如引入了 Lipton 的 $\mathrm{ELTL_{FT}}$ 规约公式等[55]. 在一系列的研究之后, 现有的 ByMC 工具已经是一个接受拓展 Promela 输入的完全自动化的验证工具. Konnov 等人[55]还对 ByMC 工具进行了并行拓展, 通过 MPI 接口实现了并行化 SMT 求解, 提高了求解速度.

2.2.3 基于 FOL 的研究工作

Berkovits 等人[40]使用基于 FOL 的建模方法和基于 SMT 的验证技术对基于阈值的共识协议 (threshold-based protocols) 进行了参数化验证. 在建模时, 将共识协议建模为用多分类 FOL 表示的状态转换系统; 在验证时, 使用 McMillan 提出的 Ivy 建模语言[56]来表达验证模型, 另外, 研究还定义了 TIP 语言 (threshold intersection property) 表达协议的共识性质. 该验证技术将验证问题分解为 EPR (effectively propositional) 逻辑和 BAPA (boolean algebra with presburger arithmetic) 逻辑两部分, 并在验证中自动生成 TIP 公式, 发送至 SMT 求解器 CVC4 和 Z3 进行求解. 该自动定理证明验证技术不受具体问题规模大小的限制, 并显示了其在建模和验证复杂协议时的有效性和灵活性. Berkovits 等人希望通过诸如并行化等方法来提高该方法的表现, 并希望在未来可以将该方法与自动推理算法相结合.

以上调研表明, 基于 SAT/SMT 的限界模型检测和自动定理证明能够比常规模型检测技术验证更大规模的共识协议, 在真实应用场景中具有更高的实用性.

2.3 基于定理证明的共识协议验证

定理证明技术通过构造公式和推论对性质做出正确性证明. 在定理证明中, 根据证明辅助工具的不同, 我们对

以下研究进行了分类. 具体可以分为: 基于 Coq 一阶逻辑的研究、基于 HOL 高阶逻辑的研究、基于证明系统的研究. 表 4 总结了基于定理证明的共识协议研究情况.

表 4 基于定理证明方法的研究

算法	验证性质	验证工具	建模思想/语言	文献	验证时间
LastVoting UniformVoting DLS	一致性 终止性[24]	数学定理证明[24]	HO模型[24]	[3,24]	—
OneThirdRule	一致性 终止性[24] 有效性[3]	数学定理证明[24], Nuprl proof assistant[3]	HO模型[24] EventML[3]		近2天[3]
Multi-Paxos	一致性 有效性	Nuprl proof assistant	EventML	[3]	近2天
	一致性	TLAPS with 3 backend checkers	TLA+	[43]	—
A(T, E) U(T, E)	一致性 终止性 有效性 不可改性	Isabelle/HOL proof assistant	HO模型	[38]	—
RAFT	状态机安全性 选举安全性 日志匹配	Coq proof assistant	Verdi framework + OCaml	[2]	—
Algorand区块链协议	区块链安全性: 异步安全性	Coq proof assistant	—	[41]	—
CKB块同步协议	可靠性 完备性	Coq proof assistant	—	[42]	—

2.3.1 基于 Coq 一阶逻辑的研究工作

一阶逻辑 (first-order logic) 也称一阶谓词演算, 是使用于数学、哲学、语言学以及计算机科学中的一种形式系统 (formal system), 可以处理简单的陈述命题, 还包含断言和量化[57]. 应用该逻辑的典型证明交互式定理证明辅助工具有 Coq 等. Coq 允许用户输入包含数学断言的表达式并帮助构造形式化证明.

广受关注的共识算法 RAFT 在 Woos 等人[2]的研究中得到了证明. Woos 等人使用 verdi framework 对 RAFT 进行建模, 并使用 Coq 工具验证了 RAFT 独有的性质: 状态机安全性、选举安全性、日志匹配和 leader 完整性, 从而证明了其安全性. 该工作对与验证高度耦合的系统不变量部分做出了优化, 从而加速了证明过程, 加强了证明的鲁棒性. Woos 等人的研究是 RAFT 共识协议的第 1 个形式化验证工作, 该工作验证了 RAFT 的安全性, 该建模验证技术为验证其他共识算法的研究提供了重要参考.

定理证明还可以用于验证区块链共识协议. Algorand 区块链协议是一个主链共识协议, 是由 proof-of-stake 协议和 BFT 协议组成的混合协议, Alturki 等人[41]的研究使用 Coq 工具对 Algorand 协议进行了形式化建模并验证, 验证的性质为异步安全性, 即两个块不会在同一轮中出块, 这是区块链的安全性性质. Alturki 等人[41]认为该模型和证明为 Algorand 协议的进一步形式化验证其他性质例如活性等提供了工作基础.

CKB 块同步协议也是 CKB 的协议之一, 它定义了区块链节点在生成新区块时如何与其他节点进行同步. Bu 等人[42]使用 Coq 工具对 CKB 块同步协议进行了形式化建模和验证, 并证明了 CKB 块同步协议的可靠性和完备性, 即在一定的可靠性和一致性假设下 CKB 块同步协议是正确的. 同时, 该工作通过消除这些假设模拟潜在的攻击, 并证明了如果没有这些假设, CKB 块同步协议将无法完成同步. Bu 等人计划在未来的研究中将该形式化验证方法运用到 CKB 的其他协议中.

2.3.2 基于 HOL 高阶逻辑的研究工作

与一阶逻辑相比, 高阶逻辑 (higher-order logic) 允许对谓词进行量化, 因此高阶逻辑的描述能力更强. Isabelle 是一个广为使用的高阶逻辑证明辅助工具, 该工具提供了一套元逻辑, 可以使用多种对象逻辑. 使用最普遍的对象

逻辑是 Isabelle/HOL. 在交互式定理证明领域, Isabelle 曾被多次用来验证共识协议.

对于 HO 体系, 最初提出 HO 模型的 Charron-Bost 等人[38]的研究中提出了通信谓词并将其与 HO 模型表达的算法结合, 进行了手工定理证明, 证明了 CFT 类的 LastVoting、UniformVoting 和 DLS 算法的一致性和终止性, 为后续基于 HO 模型的验证研究提供了理论基础[24]. Charron-Bost 等人的后续研究中[38]使用 HO 模型作为建模框架和故障模型, 对 HO 体系下的算法进行了交互式定理证明的验证, 研究结果说明 HO 模型结合模型检测技术对于 CFT 类共识协议具有较好的验证能力. 但是由于模型检测技术受限于状态空间爆炸问题, 他们选择使用 Isabelle/HOL 工具对 HO 表达的 BFT 类算法进行证明. 由于使用 HO 数学模型做了更高层的抽象, 相比于其他工作, 该工作中的交互定理证明的工作量减少了一个数量级.

2.3.3 基于证明系统的研究工作

证明系统指的是一种复合的证明工具. 该系统通常包含命题逻辑、一阶逻辑或高阶逻辑的公式作为输入, 一系列的规则用来证明定理和理论, 以及背景中假设不变的公理. Nuprl 和 TLAPS 都是常用的证明系统.

Rahli 等人[3]的研究对 HO 模型中提到的 OneThirdRule 算法进行了交互式定理证明的验证. 该研究认为模型检测要求提供实际代码, 且由于状态爆炸往往无法做到搜索全部故障场景的状态空间, 在实践方面有局限性, 因此选择使用交互式证明助手进行定理证明. 研究使用 EventML 建模语言对 OneThirdRule 进行了建模, 并使用 Nuprl proof assistant 对算法的安全性进行了验证. 这套方案后来成功地被用来实现并验证工业级的共识协议 Multi-Paxos, 证明了其可行性. Multi-Paxos 作为工业级 Paxos 的一个实现受到了不少的关注, 相比于 Lamport 的 Basic Paxos 关注对单一值投票决定一致性, Mulit-Paxos 关注对一串值达成一致. Rahli 等的研究仅讨论了安全性属性, 因此在未来 Rahli 等人计划研究 OneThirdRule 算法的非阻塞性等属性, 并且期望能够开发实现自动化工具来辅助验证.

Chand 等人[43]对 Multi-Paxos 算法的建模工作进行了总结, 发现当时还没有合适的建模语言能够对 Mulit-Paxos 算法进行建模, 主要原因在于这些语言仍然缺少对算法中具体阶段的形式化表达能力. 他们推荐使用 TLA+ 语言对 Mulit-Paxos 算法建模. 在验证部分, Chand 等人重点关注 Multi-Paxos 的安全性属性, 使用 TLAPS (TLA proof system) 进行验证, 最终高度抽象出了两条形式化建模原则 VotedInv 和 VotedOnce. 该工作展示了如何将 Lamport 对于基本 Paxos 的证明扩展到 Multi-Paxos, 进一步提出了证明复杂性质的方法, 其他共识算法尤其是 Paxos 的变体等都可以应用这两条建模原则并用定理证明方式验证算法安全性.

2.3.4 其他基于数学证明的研究工作

Bentov 等人[44]提出了 PoS 共识协议 Snow White, 逐步手工证明了协议的安全性. 研究首先描述了一个简单的理想共识协议设计, 证明了其安全性; 然后逐渐将其扩充, 使其越来越接近真实世界的共识协议, 得到一个混合协议 HYB 并证明了其安全性; 最终, 研究又证明真实世界的协议 Snow White 和混合协议 hyb 同样安全.

此外, 针对 PoS 协议, David 等人[45]基于 Ouroboros 提出了一种 PoS 类共识协议 Ouroboros Praos, 手工证明了协议在标准密码学假设下的安全性. 研究首先描述了静态情况下的协议, 逐步将其扩充, 使其成为一个处理动态情况的协议, 最终用随机预言模型证明了其安全性.

Amoussou-Guenou 等人[46]对基于拜占庭容错的 PoS 协议 Tendermint 进行了形式化分析, 证明了其在半同步系统中的正确性. 研究首先提出了一种基于轮的算法 Tendermint One-Shot 共识算法, 证明了其正确性; 此后又将其推广到一般情况, 证明了 Tendermint 协议的有效性、一致性等属性.

以上研究表明, 定理证明可以对共识协议进行验证, 且支持验证的协议种类较多, 验证能力较强. 但是, 定理证明同时也要求具备相关领域的技术和知识积累, 人工参与度高、工作量大, 对应用者的要求较高.

3 调研结论

本节在第 2 节的调研结果基础上进行总结和分析, 给出 4 个研究问题的答案.

3.1 RQ1: 在共识协议的形式化验证工作中, 哪类协议更受研究者关注?

自 Lamport 提出 Paxos 以来, 已有大量工作研究共识算法, 既有传统的分布共识协议, 又有区块链共识协议,

其种类较多. 本文对形式化验证共识协议的现有工作进行了调研, 按照 CFT 和 BFT 对算法分类, 统计了 2010 年前和 2010 年后被引量较高的文献 (引用 10 次以上) 的数量, 如图 3 所示.

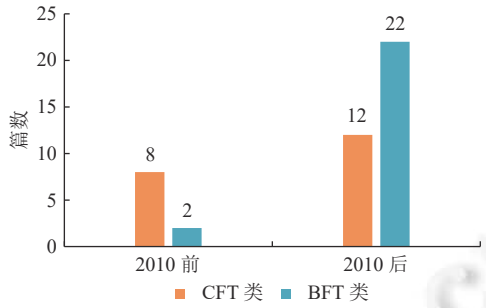


图 3 共识协议验证的研究趋势

从调研论文的整体数量上来看, 研究并验证 CFT 类和 BFT 类的高引用率文献分别有 20 篇与 24 篇. 从时间角度来看, 2010 年以前 CFT 类协议的建模与验证更受关注 (8 篇); 而在 2010 年后, 随着区块链技术的不断成熟, BFT 类协议的建模与验证更受关注 (22 篇).

在 CFT 类协议中, Paxos 及其变体算法更受关注, 如 LastVoting、DLS、Mulit-Paxos、RAFT 等. 在 BFT 类协议中, 研究者们更关注区块链协议的正确性证明, 包括 Algorand、Tendermint 等.

3.2 RQ2: 采用模型检测验证共识协议, 哪些性质容易验证, 哪些性质难以验证?

在模型检测验证共识协议的研究中, 研究者们往往会关注共识协议的一致性、有效性和终止性. 但是由于 FLP 不可能性原理, 在一个完全异步且节点会失效的分布式系统上, 终止性无法得到保证. 因此, 不能苛求终止性的条件, 验证工作一般保证严格的一致性, 并用随机化等一些方法保证一定概率的终止性, 即对终止性条件的要求做一定的放松.

图 4 展示了部分研究提供的验证技术对指定协议的性质验证的情况, 可以看出: (1) 验证有效性和不可改性的工作较少, 主要原因是有效性与不可改性验证较为简单, 不少研究工作都直接假设其成立; (2) 验证一致性和终止性的工作较多.

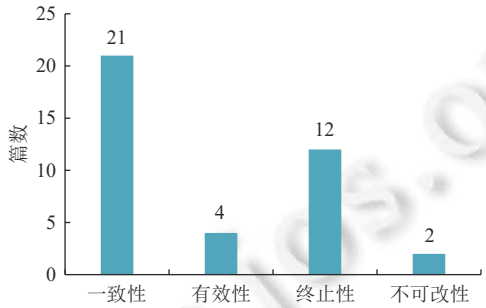


图 4 模型检测技术对共识性质的验证情况统计

图 5 统计了本文调研的文献中涉及的验证技术对特定协议一致性和终止性的验证能力, 验证能力体现为能够验证的最大集群规模. 注: 由于定理证明技术的验证能力不受集群规模限制, 在此不做统计. 图 5 中共统计了 20 组数据, 其中每组数据表示“特定共识算法-验证方法”组的验证规模. 统计表明, 所调研的 20 组模型检测技术都能够验证一致性, 但其中有 10 组技术没能成功验证终止性. 在能够同时验证一致性和终止性的研究中, 一致性比终止性的验证难度更大, 所能验证的节点规模更小. 一致性是共识协议内集群达成一致所必须保证的性质, 因此研究工作普遍关注这一性质, 未来需要进一步提升一致性的验证能力.

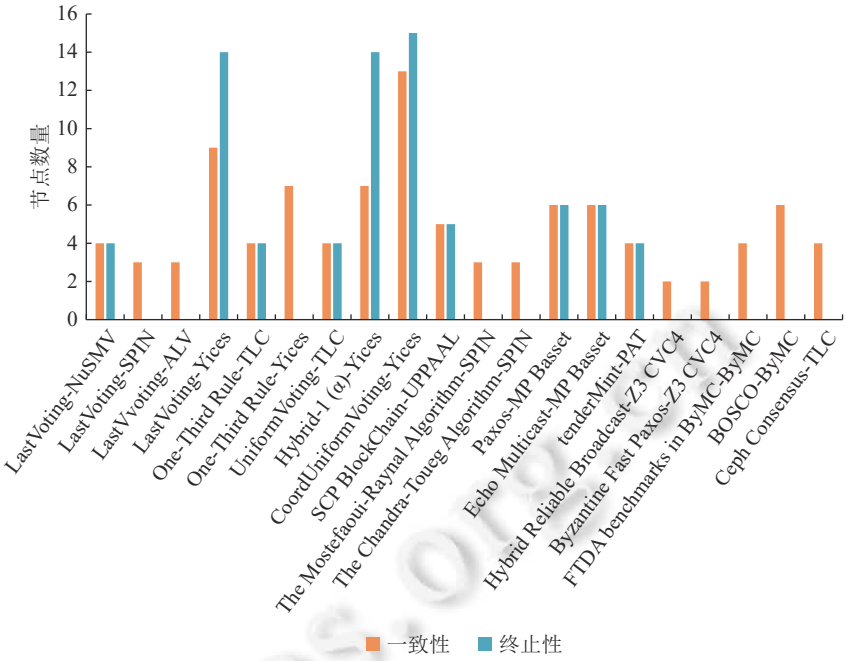


图 5　模型检测技术对一致性和终止性的验证能力统计

3.3　RQ3: 基于不同的模型检测技术验证共识协议的一致性, 哪种技术的验证能力更强?

以上调研中, 本文重点分析比对了形式化验证中的显式状态模型检测、符号模型检测、基于 SAT/SMT 的限界模型检测以及定理证明技术. 其中, 定理证明不受问题规模限制. 图 6 展示了不同模型检测技术对于一致性的验证能力, 统计了能够成功验证 3 种问题规模 (≤5 个节点、>5 且<10 个节点以及≥10 个节点) 的模型检测工作的数量.

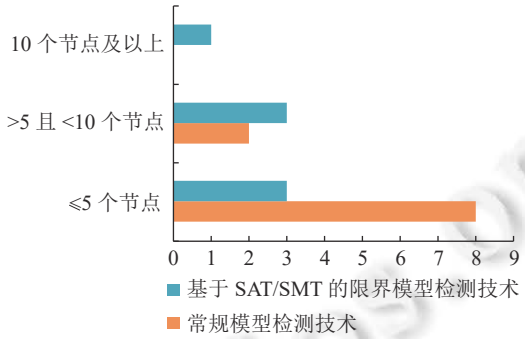


图 6　模型检测技术对一致性的验证能力

基于显式状态和符号模型检测技术的工作中, 结果最好的是 Afzaal 等人[33]使用 PAT 模型检测工具验证了自研的基于区块链的众包共识算法 STBC, 验证规模达到 9 个节点, 该算法属于 BFT 类算法. 在其他的研究中, 验证规模普遍是 3–4 个节点.

基于 SAT/SMT 的限界模型检测技术的工作中, 结果最好的是 Tsuchiya 等人的研究[25,37], 对于 CFT 类算法, 其对一致性的验证规模达到 13 个节点, 终止性的验证规模达到 15 个节点[25]. 验证 LastVoting 算法时, 使用基于 SAT/SMT 的限界模型检测, 其验证规模达到 8 个节点, 而常规模型检测技术仅能验证 4 个节点.

从验证能力来看, 基于 SAT/SMT 的限界模型检测技术对共识算法一致性的验证能力更强, 最大验证规模为 13 个节点. 影响验证规模的因素包括建模抽象度和验证优化技术. 在验证规模提升较大的工作中, 在建模方面,

HO 模型能够实现共识协议行为的高度抽象, 进而缓解状态爆炸问题; 在验证方面, 基于显式状态和符号模型检测技术, 在集群规模不大时就会遇到状态爆炸问题, 基于 SAT/SMT 的限界模型检测技术的验证能力更强.

3.4 RQ4: 共识协议的形式化验证方向未来的研究趋势是什么?

本文经过调研发现模型检测技术在共识协议验证工作的验证能力有限, 最好的情况也仅能验证 13 个节点, 因此如何提升模型检测技术对共识协议一致性的验证能力是未来研究的重点之一, 本文分别从建模和验证方面总结该问题的机遇和挑战.

(1) 面向验证的共识协议抽象建模方法. 基于 HO 模型的 CFT 类算法相比于其他工作中的建模方法取得了更好的验证效果, 主要原因在于其对协议行为进行了高度抽象, 将共识算法一致性的验证问题简化为只涉及算法执行单个阶段运行的有界模型检查问题, 在验证阶段通过计算一个阶段的运行状态, 再通过数学归纳方法, 就能验证共识算法无限阶段运行的一致性, 大幅降低了模型检验的复杂度. 对于其他 CFT 类和 BFT 类共识协议, 需要研究领域特定的抽象建模方法, 取代通用建模方法, 以提升其验证能力. 此外, 可以借鉴 HO 建模方法, 或扩展 HO 建模语言, 使其能够对其他 CFT 类和 BFT 类共识协议进行建模, 再生成对应的 SMT/SAT 模型, 调用先进的 SMT/SAT 求解器进行形式化验证.

(2) 共识协议的模型检测优化方法. 模型检测技术的优化方法可以划分为 3 类. 第 1 类是针对限界模型检测的于并行化优化技术, SAT competition 和 SMT competition 中都设置了并行竞赛环节, 支持和鼓励并行优化技术研究, 将难实例问题拆分为子问题, 派发至多求解器并行求解, 再对求解结果进行综合, 其中重点需要解决子问题的合理拆分问题. 第 2 类是构建领域验证知识知识库, 融合定理证明中的部分引理知识, 增加验证约束, 进而提升验证效率. 第 3 类是基于协议行为模式的模型检测优化方法, 本文作者前期工作中针对实时性质的模型检测验证提出了基于行为模式的状态空间压缩方法[58], 构造与原模型行为等价的压缩模型, 且压缩后模型的状态空间显著小于元模型的状态空间. 由于共识协议的行为特征明显, 未来需要解决其行为模式的识别问题和基于行为模式的共识协议模型状态空间压缩方法.

4 结 论

本文经过调研, 研究总结了共识协议的形式化验证的研究工作, 并根据其所用技术的不同将其分为 3 类: 常规模型检测技术、基于 SAT/SMT 的限界模型检测技术以及定理证明技术. 在常规模型检测技术与基于 SAT/SMT 的限界模型检测技术中, 本文研究发现, 后者的验证能力强于前者, 未来验证共识协议等的研究工作可以参考使用基于 SAT/SMT 的限界模型检测技术. 而关于模型检测技术与定理证明技术, 二者各有优缺, 模型检测技术自动化程度高, 入门成本低, 且不易出错, 但是由于状态爆炸问题导致其验证能力有限; 定理证明技术可以从数学证明的角度证明协议的正确性, 但是需要大量经验和技术积累, 应用成本高. 综合以上, 本文认为基于 SAT/SMT 的限界模型检测技术的前景更加明朗.

在调研方法方面, 本文没有对所相关的建模和验证技术进行独立验证或实验, 主要有以下 3 方面原因: (1) 目前共识协议的形式化验证研究往往针对其项目内所需要采用的共识协议提出适合这类协议的形式验证方法, 尽量达到验证结果可用的目的, 也因为如此, 很多工作难以进行横向对比; (2) 缺少横向对比的 benchmark, 目前只有 Konnov 等人[39]提出了基于 ByMC 工具的 benchmark, 用于对比多种基于模型检测的验证方法, 其中包含的共识协议并不能覆盖本研究所调研的全部共识协议; (3) 本文侧重于文献调研, 后续的研究工作中会选择其中部分共识协议进行实证研究, 并与我们提出的验证方法进行对比, 例如, 与基于 HO 语言建模且基于 SAT/SMT 验证技术的方法进行实验对比.

在未来的研究中, 一方面可以研究面向验证的共识协议抽象建模方法, 在建模中考虑并避免验证的受限因素; 另一方面可以研究共识协议的模型检测优化方法, 采用并行化验证技术充分利用高性能计算资源, 或研究如何构建共识协议验证知识和行为模式的领域知识库, 融合领域知识提升验证效率, 或基于行为模式构建状态空间压缩的替代模型.

References:

[1] Tanenbaum AS, van Steen M. Distributed Systems: Principles and Paradigms. Upper Saddle River, NJ: Prentice Hall, 2002.

[2] Woos D, Wilcox JR, Anton S, Tatlock Z, Ernst MD, Anderson TE. Planning for change in a formal verification of the raft consensus

protocol. In: Proc. of the 5th ACM SIGPLAN Conf. on Certified Programs and Proofs. St. Petersburg: ACM, 2016. 154–165. [doi: 10.1145/2854065.2854081]

[3] Rahli V, Guaspari D, Bickford M, Constable RL. Formal specification, verification, and implementation of fault-tolerant systems using EventML. In: Proc.of the 15th Int'l Workshop on Automated Verification of Critical Systems (AVoCS 2015). 2015.

[4] Lamport L, Shostak R, Pease M. The Byzantine generals problem. Concurrency: The Works of Leslie Lamport. New York: ACM, 2019. 203–226. [doi: 10.1145/3335772.3335936]

[5] Xia Q, Dou WS, Guo KW, Liang G, Zuo C, Zhang FJ. Survey on blockchain consensus protocol. Ruan Jian Xue Bao/Journal of Software, 2021, 32(2): 277−299 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6150.htm [doi: 10.13328/j.cnki.jos.006150]

[6] Lamport L. The part-time parliament. Concurrency: The Works of Leslie Lamport. New York: ACM, 2019. 277–317. [doi: 10.1145/3335772.3335939]

[7] Ongaro D, Ousterhout J. In search of an understandable consensus algorithm. In: Proc. of the 2014 USENIX Conf. on USENIX Annual Technical Conf. Philadelphia: USENIX Association, 2014. 305–320.

[8] Castro M, Liskov B. Practical Byzantine fault tolerance and proactive recovery. ACM Trans. on Computer Systems, 2002, 20(4): 398–461. [doi: 10.1145/571637.571640]

[9] Lamport L. Paxos made simple. ACM SIGACT News, 2001, 51–58.

[10] Fischer MJ, Lynch NA, Paterson MS. Impossibility of distributed consensus with one faulty process. Journal of the ACM, 1985, 32(2): 374–382. [doi: 10.1145/3149.214121]

[11] Kwiatkowska MZ, Norman G, Segala R. Automated verification of a randomized distributed consensus protocol using cadence SMV and PRISM. In: Proc. of the 13th Int'l Conf. on Computer Aided Verification. Paris: Springer, 2001. 194–206. [doi: 10.1007/3-540-44585-4_17]

[12] Bjesse P. What is formal verification? ACM SIGDA Newsletter, 2005, 35(24): 1–es.

[13] Pu L, Chen LQ, Chen Z, *et al*. Research progress and trend of formal methods. In: CCF, ed. CCF 2017–2018 China Computer Science and Technology Development Report (in Chinese). Beijing: China Machine Press, 2018.

[14] Harrison J, Urban J, Wiedijk F. History of interactive theorem proving. In: Siekmann JH, ed. Computational Logic. Handbook of the History of Logic. Amsterdam: Elsevier, 2014.

[15] Emerson EA, Clarke EM. Characterizing correctness properties of parallel programs using fixpoints. In: Proc. of the 7th Int'l Colloquium on Automata, Languages, and Programming. Noordwijkerhout: Springer, 1980. 169–181. [doi: 10.1007/3-540-10003-2_69]

[16] Clarke EM, Emerson EA. Design and synthesis of synchronization skeletons using branching time temporal logic. In: Grumberg O, Veith H, eds. 25 Years of Model Checking. Berlin: Springer, 2008. 196–215. [doi: 10.1007/978-3-540-69850-0_12]

[17] Huth MRA, Ryan M. Logic in Computer Science: Modelling and Reasoning about Systems. Cambridge: Cambridge University Press, 1999.

[18] Hou G, Zhou KJ, Yong JW, Ren LT, Wang XL. Survey of state explosion problem in model checking. Computer Science, 2013, 40(S1): 77–86, 111 (in Chinese with English abstract). [doi: 10.3969/j.issn.1002-137X.2013.z1.018]

[19] Clarke EM, Klieber W, Nováček M, Zuliani P. Model checking and the state explosion problem. In: Meyer B, Nordio M, eds. LASER Summer School on Software Engineering. Berlin: Springer, 2011. 1–30. [doi: 10.1007/978-3-642-35746-6_1]

[20] Clarke EM, Filkorn T, Jha S. Exploiting symmetry in temporal logic model checking. In: Proc. of the 5th Int'l Conf. on Computer Aided Verification. Elounda: Springer, 1993. 450–462. [doi: 10.1007/3-540-56922-7_37]

[21] Visser W, Barringer H. Memory efficient state storage in Spin. In: Proc. of the 1996 DIMACS Workshop, the Spin Verification System. New Brunswick: DIMACS/AMS, 1996. 185–203.

[22] Garavel H, Mateescu R, Smarandache I. Parallel state space construction for model-checking. In: Proc. of the 8th Int'l SPIN Workshop on Model Checking of Software. Toronto: Springer, 2001. 217–234. [doi: 10.1007/3-540-45139-0_14]

[23] Groce A, Visser W. Heuristics for model checking Java programs. Int'l Journal on Software Tools for Technology Transfer, 2004, 6(4): 260–276. [doi: 10.1007/s10009-003-0130-9]

[24] Charron-Bost B, Schiper A. The Heard-Of model: Computing in distributed systems with benign faults. Distributed Computing, 2009, 22(1): 49–71. [doi: 10.1007/s00446-009-0084-6]

[25] Tsuchiya T, Schiper A. Verification of consensus algorithms using satisfiability solving. Distributed Computing, 2011, 23(5): 341–358. [doi: 10.1007/s00446-010-0123-3]

[26] Chaouch-Saad M, Charron-Bost B, Merz S. A reduction theorem for the verification of round-based distributed algorithms. In: Proc. of the 3rd Int'l Workshop on Reachability Problems. Palaiseau: Springer, 2009. 93–106. [doi: 10.1007/978-3-642-04420-5_10]

[27] Noguchi T, Tsuchiya T, Kikuno T. Safety verification of asynchronous consensus algorithms with model checking. In: Proc. of the 18th

IEEE Pacific Rim Int'l Symp. on Dependable Computing. Niigata: IEEE, 2012. 80–88. [doi: 10.1109/PRDC.2012.24]

[28] Gao S, Zhan BH, Liu DP, Sun XC, Zhi YN, Janse ND, Zhang LJ. Formal verification of consensus in the Taurus distributed database. In: Huisman M, Păsăreanu C, Zhan NJ, eds. Proc. of the 2021 Lecture Notes in Computer Science, Formal Method (FM 2021). Cham: Springer, 2021. [doi: 10.1007/978-3-030-90870-6_42]

[29] Das Neves Fernandes A. Formal verification of the Ceph consensus algorithm using TLA+ [Ph.D. Thesis]. Universidade do Porto (Portugal). 2021. https://repositorio-aberto.up.pt/bitstream/10216/139563/2/529181.pdf

[30] Yoo J, Jung Y, Shin D, Bae M, Jee E. Formal modeling and verification of a federated byzantine agreement algorithm for blockchain platforms. In: Proc. of the 2019 IEEE Int'l Workshop on Blockchain Oriented Software Engineering (IWBOSE). Hangzhou: IEEE, 2019. [doi: 10.1109/IWBOSE.2019.8666514]

[31] Sun M, Lu YT, Feng YC, Zhang Q, Liu SY. Modeling and verifying the CKB blockchain consensus protocol. Mathematics. 2021, 9(22): 2954. [doi: 10.3390/math9222954]

[32] Thin WYMM, Dong NP, Bai GD, Dong JS. Formal analysis of a proof-of-stake blockchain. In: Proc. of the 23rd Int'l Conf. on Engineering of Complex Computer Systems (ICECCS). Melbourne: IEEE, 2018. 197–200. [doi: 10.1109/ICECCS2018.2018.00031]

[33] Afzaal H, Imran M, Janjua MU, Gochhayat SP. Formal modeling and verification of a blockchain-based crowdsourcing consensus protocol. IEEE Access, 2022, 10: 8163–8183. [doi: 10.1109/ACCESS.2022.3141982]

[34] Bokor P, Kinder J, Serafini M, Suri N. Efficient model checking of fault-tolerant distributed protocols. In: Proc. of the 41st IEEE/IFIP Int'l Conf. on Dependable Systems & Networks (DSN). Hong Kong: IEEE, 2011. 73–84. [doi: 10.1109/DSN.2011.5958208]

[35] Drăgoi C, Henzinger TA, Veith H, Widder J, Zufferey D. A logic-based framework for verifying consensus algorithms. In: Proc. of the 15th Int'l Conf. on Verification, Model Checking, and Abstract Interpretation. San Diego: Springer, 2014. 161–181. [doi: 10.1007/978-3-642-54013-4_10]

[36] Minamikawa T, Tsuchiya T, Kikuno T. Towards automated verification of distributed consensus protocols. In: Proc. of the 16th Asia-Pacific Software Engineering Conf. Batu Ferringhi: IEEE, 2009. 499–506. [doi: 10.1109/APSEC.2009.23]

[37] Tsuchiya T, Schiper A. Using bounded model checking to verify consensus algorithms. In: Proc. of the 22nd Int'l Symp. on Distributed Computing. Arcachon: Springer, 2008. 466–480. [doi: 10.1007/978-3-540-87779-0_32]

[38] Charron-Bost B, Debrat H, Merz S. Formal verification of consensus algorithms tolerating malicious faults. In: Proc. of the 13th Int'l Conf. on Stabilization, Safety, and Security of Distributed Systems. Grenoble: Springer, 2011. 120–134. [doi: 10.1007/978-3-642-24550-3_11]

[39] Konnov I, Veith H, Widder J. On the completeness of bounded model checking for threshold-based distributed algorithms: Reachability. Information and Computation, 217, 252: 95–109. [doi: 10.1016/j.ic.2016.03.006]

[40] Berkovits I, Lazić M, Losa G, Padon O, Shoham S. Verification of threshold-based distributed algorithms by decomposition to decidable logics. In: Proc. of the 31st Int'l Conf. on Computer Aided Verification. Los Angeles: Springer, 2019. 245–266. [doi: 10.1007/978-3-030-25543-5_15]

[41] Alturki MA, Chen J, Luchangco V, Moore B, Palmskog K, Peña L, Rosu G. Towards a verified model of the algorand consensus protocol in Coq. In: Proc. of the 3rd Int'l Symp. on Formal Methods. Porto: Springer, 2019. 362–367. [doi: 10.1007/978-3-030-54994-7_27]

[42] Bu H, Sun M. Towards Modeling and Verification of the CKB Block Synchronization Protocol in Coq. In: Lin SW, Hou Z, Mahony B, eds. Proc. of the 2020 Formal Methods and Software Engineering (ICFEM 2020). Cham: Springer, 020. 287–296. [doi: 10.1007/978-3-030-63406-3_17]

[43] Chand S, Liu YA, Stoller SD. Formal verification of Multi-Paxos for distributed consensus. In: Proc. of the 21st Int'l Symp. on Formal Methods. Limassol: Springer, 2016. 119–136. [doi: 10.1007/978-3-319-48989-6_8]

[44] Bentov I, Pass R, Shi E. Snow White: Provably secure proofs of stake. IACR Cryptol. ePrint Arch., 2016.

[45] David BM, Gazi P, Kiayias A, Russell A. Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. IACR Cryptol. ePrint Arch., 2017.

[46] Amoussou-Guenou Y, Del Pozzo A, Potop-Butucaru M, Piergiovanni ST. Correctness of tendermint-core blockchains. In: Proc. of the 22nd Int'l Conf. on Principles of Distributed Systems (OPODIS 2018). Hong Kong: Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2018. 16:1–16:16.

[47] Konnov I, Veith H, Widder J. SMT and POR beat counter abstraction: Parameterized model checking of threshold-based distributed algorithms. In: Proc. of the 27th Int'l Conf. on Computer Aided Verification. San Francisco: Springer, 2015. 85–102. [doi: 10.1007/978-3-319-21690-4_6]

[48] Gafni E. Round-by-round fault detectors (extended abstract): Unifying synchrony and asynchrony. In: Proc. of the 17th Annual ACM Symp. on Principles of Distributed Computing. Puerto: ACM, 1998. 143–152. [doi: 10.1145/277697.277724]

[49] Santoro N, Widmayer P. Agreement in synchronous networks with ubiquitous faults. Theoretical Computer Science, 2007, 384(2−3): 232–249. [doi: 10.1016/j.tcs.2007.04.036]

[50] Mazières D. The stellar consensus protocol: A federated model for internet level consensus. 2018. https://www.stellar.org/cn/papers/stellar-consensusprotocol.pdf

[51] Biely M, Widder J, Charron-Bost B, Gaillard A, Hutle M, Schiper A. Tolerating corrupted communication. In: Proc. of the 26th Annual ACM Symp. on Principles of Distributed Computing. Portland: ACM, 2007. 244–253. [doi: 10.1145/1281100.1281136]

[52] John A, Konnov I, Schmid U, Veith H, Widder J. Towards modeling and model checking fault-tolerant distributed algorithms. In: Proc. of the 20th Int'l SPIN Workshop on Model Checking of Software. Stony Brook: Springer, 2013. 209–226. [doi: 10.1007/978-3-642-39176-7_14]

[53] John A, Konnov I, Schmid U, Veith H, Widder J. Parameterized model checking of fault-tolerant distributed algorithms by abstraction. In: Proc. of the 2013 Formal Methods in Computer-aided Design. Portland: IEEE, 2013. 201–209. [doi: 10.1109/FMCAD.2013.6679411]

[54] Gmeiner A, Konnov I, Schmid U, Veith H, Widder J. Tutorial on parameterized model checking of fault-tolerant distributed algorithms. In: Proc. of the 14th Int'l School on Formal Methods for the Design of Computer, Communication and Software Systems. Cham: Springer, 2014. 122–171. [doi: 10.1007/978-3-319-07317-0_4]

[55] Konnov I, Lazić M, Veith H, Widder J. A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms. In: Proc. of the 44th ACM SIGPLAN Symp. on Principles of Programming Languages. Paris: ACM, 2017. 719–734. [doi: 10.1145/3009837.3009860]

[56] McMillan KL, Padon O. Deductive verification in decidable fragments with Ivy. In: Proc. of the 25th Int'l Static Analysis Symp. Freiburg: Springer, 2018. 43–55. [doi: 10.1007/978-3-319-99725-4_4]

[57] Rautenberg W. A Concise Introduction to Mathematical Logic. New York: Springer, 2006. [doi: 10.1007/0-387-34241-9]

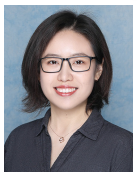
[58] Ge N, Pantel M. Real-time property specific reduction for time Petri net. In: Proc. of the 2014 Int'l Workshop on Petri Nets and Software Engineering, Co-located with the 35th Int'l Conf. on Application and Theory of Petri Nets and Concurrency (PetriNets 2014) and the 14th Int'l Conf. on Application of Concurrency to System Design. Tunis: CEUR-WS.org, 2014. 165–179.

附中文参考文献:

[5] 夏清, 窦文生, 郭凯文, 梁赓, 左春, 张凤军. 区块链共识协议综述. 软件学报, 2021, 32(2): 277−299. http://www.jos.org.cn/1000-9825/6150.htm [doi: 10.13328/j.cnki.jos.006150]

[13] 卜磊, 陈立前, 陈哲, 等. 形式化方法的研究进展与趋势. 见: 中国计算机学会, 编. CCF 2017–2018中国计算机科学技术发展报告. 北京: 机械工业出版社, 2018.

[18] 侯刚, 周宽久, 勇嘉伟, 任龙涛, 王小龙. 模型检测中状态爆炸问题研究综述. 计算机科学, 2013, 40(S1): 77–86, 111. [doi: 10.3969/j.issn.1002-137X.2013.z1.018]



葛宁(1983－), 女, 副教授, 博士生导师, CCF 专业会员, 主要研究领域为形式化方法, 模型驱动, 智能化软件工程.

李晓洲(1999－), 男, 本科生, 主要研究领域为形式化方法.

贺俞凯(1997－), 男, 硕士生, 主要研究领域为模型驱动, 形式化方法.

张莉(1968－), 女, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为模型驱动, 软件工程.

翟树茂(1998－), 男, 硕士生, 主要研究领域为形式化方法, 机器学习.