Accepted: 08 December 2025
Revised: 04 December 2025
Received: 26 August 2025

RESEARCH-ARTICLE

A Brief History of Formal Methods in China

NAIJUN ZHAN, Peking University, Beijing, China

JIM C P WOODCOCK, Southwest University, Chongqing, China

JI WANG, National University of Defense Technology China, Changsha, Hunan, China

MINGSHUAI CHEN, Zhejiang University, Hangzhou, Zhejiang, China

A Brief History of Formal Methods in China

NAIJUN ZHAN, School of Computer Science, Peking University, Beijing, China and Zhongguancun Laboratory, Beijing, China
JIM WOODCOCK, Southwest University, Chongqing, China, Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark, and Computer Science, University of York, York, United Kingdom of Great Britain and Northern Ireland
JI WANG, State Key Laboratory for Complex & Critical Software Environment, College Computer Science and Technology, National University of Defense Technology, Changsha, China
MINGSHUAI CHEN, College of Computer Science and Technology, Zhejiang University, Hangzhou, China

The development of formal methods (FM) in China dates back to the early 1950s, when several logicians shifted their research focus from mathematics to theoretical computer science and began advocating the application of mathematical logic to enhance the rigor of computing systems. A significant expansion of FM in China emerged in the 1980s, pioneered by a new generation of talented computer scientists who had visited, studied, and/or worked in Western countries, such as the United Kingdom and the United States, closely tied to China's reform and opening-up policy. A notable milestone was the establishment of the United Nations University International Institute for Software Technology (UNU/IIST) in Macau in the early 1990s, which played a crucial role in advancing FM research and collaboration in China. In recent years, the return of an increasing number of talented young scholars has further strengthened China's FM community, elevating its influence and contribution within the global FM landscape.

CCS Concepts: • **General and reference → Surveys and overviews**; • **Software and its engineering → Formal methods**; • **Theory of computation → Logic and verification**; **Automated reasoning**.

Additional Key Words and Phrases: Formal methods, duration calculus, unifying theories of programming, process algebra, theorem proving, model checking

1 Introduction

Formal methods are mathematically rigorous techniques for specifying, developing, analyzing, and verifying software and hardware systems [17]. The notion of *formal methods*—or more precisely, the *principle of formalization*—can be traced back to the renowned Hilbert's Program [118, 353] of the 1920s, which sought to establish a complete, consistent, and decidable axiomatic foundation for *all* mathematics. Hilbert's Program left few enduring legacies after Gödel's incompleteness theorems [97], but one that remains is the principle of

This article reflects the authors' perspectives and primarily focuses on the development of formal methods in Mainland China. We apologize for any significant results or contributors that may have been unintentionally omitted.

Authors' Contact Information: Naijun Zhan, School of Computer Science, Peking University, Beijing, Beijing, China and Zhongguancun Laboratory, Beijing, Beijing, China; e-mail: njzhan@pku.edu.cn; Jim Woodcock, Southwest University, Chongqing, Chongqing, China and Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark and Computer Science, University of York, York, England, United Kingdom of Great Britain and Northern Ireland; e-mail: jim.woodcock@york.ac.uk; Ji Wang, State Key Laboratory for Complex & Critical Software Environment, College Computer Science and Technology, National University of Defense Technology, Changsha, Hunan, China; e-mail: wj@nudt.edu.cn; Mingshuai Chen, College of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang, China; e-mail: m.chen@zju.edu.cn.

ACM 1433-299X/2025/12-ART
https://doi.org/10.1145/3783996

formalization: mathematicians systematically use formal languages to express statements and manipulate them according to well-defined syntactic rules. Researchers subsequently developed this principle into a foundational framework that underpins many of the fundamentals of theoretical computer science. A broad spectrum of these fundamentals—particularly logic calculi, formal languages, automata theory, and program semantics—collectively forms what is now known as formal methods (FM). Engineers have since extensively employed these methods to ensure the reliability and effectiveness of safety-critical systems across domains such as aerospace, transportation, healthcare, and defence.

The early FM development followed two complementary trajectories: The *theoretical perspective*—as pioneered by Church [54] and Turing [253] during the 1930s–1940s—aimed to establish a mathematical foundation for computation and programming, while the *engineering perspective*—as initiated by the NATO Software Engineering Conferences in the late 1960s—focused on enforcing rigorous quality assurance in software development. Together, these two lines of development gave rise to a substantial body of influential contributions, including, but not limited to, the work of eighteen ACM A. M. Turing Award laureates (see [267]). Amid this global landscape, formal methods research in China emerged in the early 1950s, when several logicians redirected their focus from mathematics to theoretical computer science and began promoting the use of mathematical logic to enhance the rigour of computing systems. Since then, FM research in China has undergone steady evolution and experienced remarkable growth and innovation. We can broadly categorize its overall development into three major phases: the *infant stage* (1950s–1970s), the *early development stage* (1980s–1990s), and the *expansion and maturity stage* (since 2000s).

In this article, we provide a historical account of the development of formal methods in China. In Sections 2 to 4, we review key contributions and landmark projects across the three major stages mentioned above. In Section 5, we examine the contributions and applications in industrial contexts. Section 6 identifies several key challenges and promising future directions in terms of research, application, and talent cultivation. We provide the concluding remarks in Section 7.

2 Infant Stage (1950s–1970s)

The development of formal methods in China dates back to the early 1950s, pioneered by three mathematicians: Shihua Hu (28 January 1912 – 11 April 1998), Xianghao Wang (5 May 1915 – 4 May 1993), and Shaokui Mo (13 August 1917 – 14 October 2011).

Shihua Hu studied mathematical logic at the University of Vienna and finally obtained his PhD on mathematical logic from the University of Münster [123]. After returning from Europe in 1943, Hu worked at Sun Yat-sen University, later moved to Central University (renamed Nanjing University in 1952), and subsequently joined Peking University. In 1950, he joined the Institute of Mathematics of the Chinese Academy of Sciences (IMCAS), and initiated the division of theoretical computer science there. Notably, since the 1950s, he has established an annual training school in theoretical computer science and mathematical logic at IMCAS for Chinese scholars, and many of these trainees have gone on to become academic leaders in the development of FM in China. Later, he and the whole division moved to the Institute of Computing Technology of the Chinese Academy (ICTCAS), which the Academy had established at the end of the 1950s after separating it from IMCAS. After joining ICTCAS, the division strengthened and formed nine research groups that focus on algorithms and complexity, computer science logic, computational models, formal languages and automata, networking, system software, and other related areas. This development marks the starting point of theoretical computer science in China. In 1985, the division separated from ICTCAS and founded the Institute of Software, Chinese Academy of Sciences (ISCAS). Since then, ISCAS has taken a leading role in developing formal methods in China, particularly in the early stages. Scholars regard Hu as the founder of mathematical logic and theoretical computer science in China.

Xianghao Wang obtained his PhD in algebra from Princeton University in 1946. He then returned to China and became a professor at Peking University. In the 1950s, he joined Jilin University and initiated the Mathematics Department there. Meanwhile, he shifted his interests to computer science, logic, theorem proving, and artificial intelligence (AI), and thereby initiated the Computer Science Department at Jilin University in the 1970s.

Shaokui Mo studied mathematical logic in EPFL and at Paris University under the supervision of Paul Bernays from 1947 to 1950. Then, he joined Nanjing University as a professor of mathematics. His research interests span a wide range of topics in mathematical logic, with a particular focus on multiple-value logics. Likewise, in the early 1950s, he realized that mathematical logic plays a foundation for Computer Science, and guided some of his students to theoretical computer science. In 1978, the Computer Science Department at Nanjing University was established as a separate entity from the Mathematics Department. Since then, Nanjing University has become one of the major institutions specializing in formal methods in China.

3 Early Development (1980s–1990s)

3.1 Early Stage of Reform and Opening-up

In the early 1980s, as part of China's reform and opening-up policy, some talented Chinese computer scientists were sent to Western countries to learn advanced science and high technologies, including theoretical computer science. In what follows, we list a few representatives.

Firstly, several pioneering individuals from ISCAS have cultivated the use of formal methods in China. Zhisong Tang completed his undergraduate and graduate studies in philosophy at Tsinghua University, then joined Renmin University, and later moved to IMCAS, where he worked in the mathematical logic division headed by Shihua Hu. From 1979 to 1981, he visited Stanford University, where he proposed executable temporal logic. After returning, he developed the executable temporal logic language XYZ/E and the corresponding software tools [245], which can express programs and their specifications in a unified framework. Yunmei Dong earned his Bachelor's degree in mathematics from Jilin University, then joined ICTCAS, and later moved to ISCAS when it was founded. He visited Stanford in 1978. He investigated recursion theory over words and its application to programming languages [65, 66] since the early 1960s, a line of work closely related to LISP. Chaochen Zhou obtained his Bachelor's degree in mathematics from Peking University and became the first graduate student of Shihua Hu in ICTCAS in 1963. He visited Tony Hoare from 1979 to 1981, and proposed axiomatic semantics for Communicating Sequential Processes (CSP) during the visit [385]. Later, he established Duration Calculus (DC) jointly with Tony Hoare and Anders Ravn [390], which provided a novel approach to the formal design of real-time systems. Ren-Ji Tao is another student of Shihua Hu, who obtained his Bachelor's degree in mathematics from Peking University. He proposed a theory of invertible finite state automata over a ring and its application to cryptography [246]. Huimin Lin obtained his PhD from ISCAS in 1986 under the supervision of Zhisong Tang, focusing on algebraic semantics. Afterwards, he worked as a research fellow at the University of Edinburgh. He proposed the theory of symbolic bisimulation in process algebra jointly with Mathew Hennessy [116, 180].

Peking University is another institution playing an essential role in the development of FM in China. Xiwen Ma initially led theoretical computer science at Peking University. He obtained his Bachelor's in mathematics from Peking University. He and Chaochen Zhou are classmates during their undergraduate studies at Peking University. He visited McCarthy at Stanford from 1978 to 1981. He initiated Theoretical Computer Science Institute at Peking University after he returned from the US. His interests are diverse, spanning mathematics, computer science, languages, literature, and music. His contributions to FM include a relational approach to formal semantics [203], the formal semantics of Prolog, and the modal logic of knowledge [302], among others.

As mentioned in the previous section, influenced by Shaokui Mo, Nanjing University also plays a vital role in the development of FM in China, even in the early stages. Jiafu Xu was considered a pioneer in software, particularly in programming theories and software synthesis [23]. Jiafu Xu obtained his Bachelor's degree in mathematics from Central University (renamed to Nanjing University in 1952) in 1948. He worked there afterwards until his passing in 2018. From 1957 to 1959, he visited the Soviet Union for two years and began working on programming theories during that time. He designed programming languages (XCY etc.), specification languages (FGSPEC etc.), quantum programming languages (NDQFP etc.), and developed several corresponding automated systems, such as XCY-compiler, NDADAS, NDSAIL, etc. [305–308]. He also proposed an analogy calculus and the analogical program derivation method [197, 332]. Zhongxiu Sun is another important person from Nanjing University. He also obtained his Bachelor's degree in mathematics from Nanjing University in 1957 and worked there until his passing in 2013. At an early stage of his career, he focused mainly on mathematical logic under the guidance of Shaokui Mo. From 1965 to 1967, he visited the UK. Afterwards, he shifted his research interests to computer science, particularly operating systems and algorithms. In 1979, he visited the University of Wisconsin and began working on the theory and implementation of distributed computing systems. Additionally, it is worth mentioning that in 1984, Yongsen Xu visited Dines Bjørner at the Technical University of Denmark and subsequently proposed an executable specification language based on Vienna Development Method (VDM) and data abstraction [132, 133]; In 1993, Jian Lu visited Cliff Jones at the University of Manchester in the UK and later introduced data decomposition into VDM [196].

Led by Huowang Chen, the National University of Defense Technology is one of the top software research and development bases in China. Huowang Chen took three years to obtain his Bachelor's degree in Mathematics from Fudan University in 1956. From 1956 to 1959, he studied Mathematical Logic under the chairmanship of Shihua Hu at Peking University. As one of the early visitors to Europe in computer science, he learned programming at the UK's National Physical Laboratory from 1965 to 1967. Huowang Chen led the first design and implementation of the Fortran compiler for 441B-series computers in China. He is also the chief designer of the software systems for the YH-1 Supercomputer, the first Petaflops supercomputer in China. His research interests include software methodology and formal methods, particularly programming languages and their implementation, as well as systems. Huowang Chen and his students have made significant contributions to formal semantics and formal verification of software. A notable 1986 work on the formal semantics of CSP proved that the equivalence between infinite streams and finite observation semantics holds. His group contributed to the formal semantics of AI Systems [278] [260], formal verification of reactive systems [261] [262] [168], a higher-order unification algorithm for theorem proving [239] and so on. Huowang Chen has actively promoted the education, research, and development of formal methods in China. He is also the founding chair of CCF-TCTCS (Technical Committee on Theoretical Computer Science) of the China Computer Foundation (CCF).

Beihang University is another vital university on FM due to its former rector, Wei Li. Wei Li obtained his PhD from the University of Edinburgh under the supervision of Gordon Plotkin on structural operational semantics [153] in 1983. After returning to China, he began lecturing at Beihang University and earned promotion to full professor in 1986. After returning, he mainly focused on type-theoretic approaches to program development [154], later shifting to specification logics for knowledge and proposing well-known logics, including Open Logic [155] and R-Calculus [156].

In the early stage of FM in China, Yongqiang Sun, a professor from Shanghai Jiaotong University, also played a vital role. Yongqiang Sun earned a Bachelor's degree in electrical engineering from Shanghai Jiaotong University and served as a lecturer at Tsinghua University. He later joined Shanghai Jiaotong University as an associate professor and subsequently advanced to the rank of professor. He primarily focuses on programming-theoretic results, with an emphasis on functional programming and program synthesis. Influenced by him, a powerful FM team has been in place since the 1990s.

3.2 Establishment of UNU/IIST

In 1992, with financial support from the Chinese government, the Portuguese government, and the local Macau government, the United Nations University established the International Institute for Software Technology (UNU/IIST) in Macau as part of UNESCO. The mission of UNU/IIST is to conduct research on software technologies and to train IT talent for developing countries. The founding director was Dines Bjørner, and Chaochen Zhou was the vice director and a principal research fellow. At the beginning, UNU/IIST consisted of two research teams, one led by Chaochen Zhou on Duration Calculus (DC) [388, 390], and the other led by Dines Bjørner on formal software engineering, specifically, approaches based on the Rigorous Approach to Industrial Software Engineering (RAISE) [209]. In 1997, Dines Bjørner left UNU/IIST, and Chaochen Zhou took up the director position; meanwhile, Chris George joined UNU/IIST and led a group of formal software engineers. Later, in 1998, Jifeng He joined UNU/IIST and led the group of DC, and widened the research topics to the semantic foundations of programming and software engineering, like the refinement calculus for Component and Object Systems (rCOS) [113]. Around 2007, Jifeng He left UNU/IIST and joined East China Normal University (ECNU), and Zhiming Liu assumed leadership of the group. At nearly the same time, Chris George left UNU/IIST, and Tomasz Janowski assumed leadership of the group, shifting its research focus to e-Government [62]. From 2002 to 2009 (two terms), Jian Lu was on the UNU/IIST board on behalf of the Chinese government. Since 2014, UNU dismissed the two research teams, restructured UNU/IIST, and renamed it UNU Macau with a new mission to collaboratively explore how digital technologies can be leveraged for sustainable development, addressing key issues in the UN 2030 Agenda for Sustainable Development[1].

UNU/IIST had a profound impact on FM in China, markedly accelerating the nation's development in this domain:

- Firstly, many talented young Chinese scholars have visited UNU/IIST, even completing their PhD theses there, to receive well-rounded training in FM. In particular, instructors can quickly guide them to the cutting edge of FM. Many of them took the leading role in the development of FM in China after returning, for example, Jian Lv (the rector of Nanjing University from 2018 to 2022), Xuandong Li (the director of the Computer Science Department of Nanjing University from 2006 to 2018) and Jianhua Zhao (the current vice dean of School of Computer Science of Nanjing University) from Nanjing University, Ji Wang (the former director of the CCF FM technical committee), Xiaoguang Mao (the previous vice dean of College of Computer Science and Technology of National University of Defense Technology) and Zhenbang Chen from National University of Defense Technology, Naijun Zhan (the current director of the CCF FM technical committee) from ISCAS, Shengchao Qin (the director of the Fermat Labs of Huawei in Hong Kong) and Geguang Pu (the dean of the School of Software Engineering of ECNU) from Peking University, Huibiao Zhu from East China Normal University, and so on.
- Secondly, close collaborations between Chinese institutions and UNU/IIST fostered several renowned research teams in China. For example, because of cooperation with Chaochen Zhou, several strong research teams on DC at that moment, including Ji Wang's team in National University of Defence Technology, Xuandong Li's team in Nanjing University, Miaomiao Zhang's team in Tongji University, and Naijun Zhan's team in ISCAS, etc.; also, based on the collaboration with Jifeng He, ECNU built a significant group on formal methods, many of them working on programming theories, specifically UTP. Jifeng He returned to ECNU and took up the position of dean of the School of Software Engineering after leaving UNU/IIST in 2007.
- Lastly, UNU/IIST opened a window for academic exchange between the Chinese FM community and the FM international community. UNU/IIST organized various FM schools in China. For example, at the autumn FM school in Beijing in 1997, the organizers invited leading FM experts to serve as tutors, including John

[1]See https://unu.edu/macau for details.

Rushby, Jim Woodcock, and Jifeng He. In addition, UNU/IIST initiated some international conferences, like the International Colloquium on Theoretical Aspects of Computing (ICTAC) and the International Conference on Software Engineering and Formal Methods (SEFM), both by Zhiming Liu. At the same time, he worked at UNI/IIST, which provides venues for academic exchange and collaboration between the Chinese FM community and the international FM community.

3.3 Mathematics Mechanization

In China, automated theorem proving is also called *mathematics mechanization*, a term coined by Wen-tsun Wu [252]. Wen-tsun Wu is a famous mathematician in topology. During the Cultural Revolution, he shifted his research interests to theorem proving, specifically solving polynomial equations and inequalities, i.e., semi-algebraic systems (SAS). He invented the so-called *Ritt-Wu methods* [251]. Together with Gröbner bases methods [16], *Ritt-Wu* method has become one of the two primary methods for solving algebraic systems. Following Wen-tsun Wu's research line, Chinese mathematicians and computer scientists produced many vital works—for example, Jiawei Hong developed geometric theorem proving by instance [121]; Jingzhong Zhang, Xiaoshan Gao, and Shang-Ching Chou worked on point elimination and readable proof generation [52, 365]; Jingzhong Zhang and Lu Yang advanced numerical parallel theorem proving [364]; and Lu Yang and Bican Xia contributed to real root isolation and classification of parametric SAS [296, 297, 328].

3.4 Duration Calculus

Duration Calculus is due to Zhou, Hoare and Ravn [390], as one of the significant academic achievements of the ESPRIT project ProCoS (short for Provably Correct Systems) [12, 112]. DC is an extension of Interval Temporal Logic (ITL) [207] by introducing the notion of duration, which is the integral of a state function over a reference interval. DC can effectively specify and reason about qualitative properties and is therefore widely and successfully applied in the design of real-time systems, e.g., gas burners [220], the European Train Control System (ETCS) [212], and others. In the early 1990s, an influential research group led by Chaochen Zhou established the DC research centre at the Technical University of Denmark (DTU). In the early development of DC, many vital results on DC were obtained there, e.g., a complete proof system for DC due to Michael Hansen and Chaochen Zhou [103], extended DC due to Chaochen Zhou, Anders Ravn, and Michael Hansen [393], decidability and undecidability of DC due to Chaochen Zhou, Michael Hansen, and Peter Sestoft [389], a probabilistic extension of DC due to Zhiming Liu et al. [195], and so on. With Chaochen Zhou's move to UNU/IIST, the institute evolved into a new hub for DC research [192], in parallel with efforts in Mainland China. Researchers obtained many vital results in the later development of DC: e.g., a mean-value calculus due to Xiaoshan Li and Chaochen Zhou [392], probabilistic DC due to Dang Van Hung and Chaochen Zhou [128], Neighbourhood Logic due to Rana Barua, Suman Roy, and Chaochen Zhou [7], higher-order DC due to Naijun Zhan, Chaochen Zhou, and Dimitar Guelev [358, 387], DC with infinite interval due to Xiaoshan Li and Chaochen Zhou [391], model-checking of DC due to Xuandong Li et al. [4, 157, 386], the design of hybrid systems with DC, as proposed by Ji Wang et al. [351, 394], and others.

3.5 Unifying Theories of Programming

Inspired by the grand unified theory in physics—which seeks to unify disparate physical theories—Tony Hoare and Jifeng He developed the Unifying Theories of Programming (UTP) [120]. UTP builds on first-order relational calculus and interprets different kinds of programs as a specific class of predicates, called *(guarded) designs*, which satisfy a set of healthiness conditions. Its semantics interprets program constructs as operators over these (guarded) designs, while refinement corresponds to logical implication between them. UTP employs Galois connections [69] to link and translate between different theories.

After Jifeng He moved to UNU/IIST, the institute — along with Mainland China — became a major research centre for UTP. Based on UTP, Zhiming Liu, Xiaoshan Li, Jifeng He, and other collaborators, mainly from China, established a semantic foundation for component-based methods, called rCOS [47, 50, 113], as well as a toolkit for rCOS [138]. Meanwhile, UTP was applied to define formal semantics and verification of different programming languages, e.g., for the hardware description language Verilog due to Huibiao Zhu and Jifeng He [115], for Circus (a formal language that combines CSP and the state-based formal method Z [231]) due to Adnan Sherif and Jifeng He [226], for hardware/software codesign [216, 218], for UML due to Zhiming Liu, Jifeng He, Xiaoshan Li, and Yifeng Chen [194], and for the combination of CSP and DC based on UTP due to Jifeng He [109]. Jifeng He also considered extending UTP to a probabilistic setting [111]. Xiong Xu et al. considered extending UTP to cyber-physical systems and established Higher-order UTP (HUTP) [315]. Researchers have applied HUTP to define formal semantics for the Architecture Analysis & Design Language (AADL) [313], Simulink/Stateflow [317], and their combination [316]. They have also used it to demonstrate the correctness of the transformation from graphical models to formal models in SystemC and ANSI C within MARS [315, 316].

3.6 Concurreny Theories

In the 1990s, in addition to the mechanization of mathematics, DC and programming theories, such as UTP and process algebra, were also hot topics in the Chinese FM community. Huimin Lin worked with Robin Milner on process algebra, particularly with Matthew Hennessy, to propose the theory of symbolic bisimulation for the Calculus of Communicating Systems (CCS) [117], and developed a tool for symbolic bisimulation called PAM (short for Process Algebra Manipulator) [180, 181]. Later, he extended the notion of symbolic bisimulation to π-calculus [182]. Mingsheng Ying revisited process algebra from a topological viewpoint [340] and also investigated it in the probabilistic setting [341]. Yuxi Fu finished his PhD from Manchester University, and then joined Shanghai Jiaotong University, working in Yongqiang Sun's group. Later, he became the director of the Computer Science Department. He proposed χ-calculus, which is a variant of π-calculus [84]. Additionally, some work related to Petri nets, a notable example being that of Chongyi Yuan from Peking University [352].

4 Expansion and Maturity (2000s–Present)

At the end of the last century and the beginning of the new century, Tony Hoare, Jim Woodcock, and others began advocating for a Grand Challenge in computing research, specifically in the area of verifying compilers [119]. Meanwhile, influenced by Tony Hoare, Microsoft invested more attention in formal methods and in its Chinese branch. Excitingly, Microsoft has achieved numerous remarkable successes in applying FM to ensure the correctness of its software products, particularly drivers. On the other hand, in China, with the rapid development of the economy, computing technology has been increasingly used in safety-critical domains, such as high-speed trains, spacecraft, aircraft, nuclear reaction control, and so on. The Yongwen train crash [83] and other catastrophic accidents raise concerns about guaranteeing the trustworthiness of these systems, particularly their software. This national strategy demands more powerful and practical FM techniques and tools. These stimulated substantial growth in research groups and national-level projects. Since the beginning of this century, the Chinese government has invested increasing amounts of funding in basic research on the foundations of software. For example, the National Natural Science Foundation of China (NSFC) launched the Grand Research Program on Fundamentals of Trustworthy Software [114] from 2007 to 2016, led by Jifeng He; The Ministry of Science and Technology (MOST) launched key projects on high trustworthy software and library in 2008, and platform towards cyber-physical systems in 2011, and a key specific theme on advanced computing and novel software from 2021 to 2025. With this strong support, FM in China has expanded significantly, particularly in the number of research groups, the successful applications of FM in industry, and the publication of high-quality papers, all of which have increased dramatically over the past twenty years. The research and application of

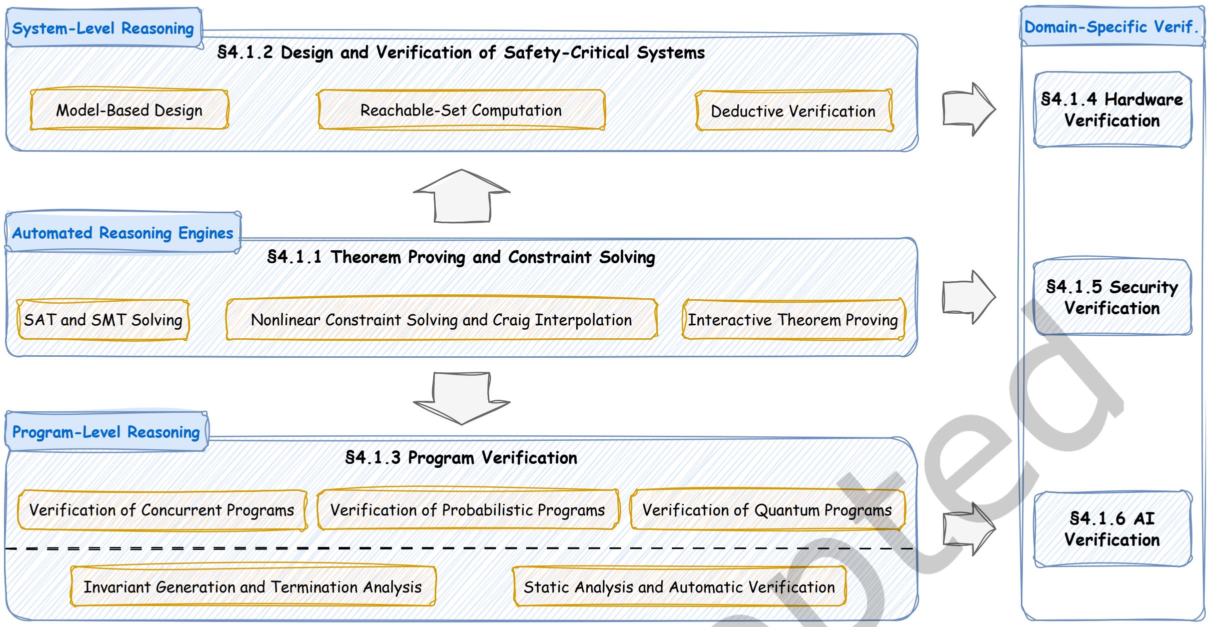
System-Level Reasoning
§4.1.2 Design and Verification of Safety-Critical Systems
Model-Based Design
Reachable-Set Computation
Deductive Verification
Automated Reasoning Engines
§4.1.1 Theorem Proving and Constraint Solving
SAT and SMT Solving
Nonlinear Constraint Solving and Craig Interpolation
Interactive Theorem Proving
Program-Level Reasoning
§4.1.3 Program Verification
Verification of Concurrent Programs
Verification of Probabilistic Programs
Verification of Quantum Programs
Invariant Generation and Termination Analysis
Static Analysis and Automatic Verification
Domain-Specific Verif.
§4.1.4 Hardware Verification
§4.1.5 Security Verification
§4.1.6 AI Verification

Fig. 1. The hierarchy for categorizing key developments of formal methods in China (2000s–present).

formal methods in China have matured and become a major contributor to the global FM community, as reflected in the growing number of papers published annually in top-tier journals and conferences in the field.

4.1 Key Developments

In this subsection, we provide a brief overview of key developments of formal methods in China over the past 20 years. We highlight results that have, according to the authors' perspectives, had a notable international impact and/or become characteristic of the Chinese FM community, either in theory or practice. These results are categorized according to the hierarchy depicted in Section 1.

4.1.1 Theorem Proving and Constraint Solving.

SAT and SMT Solving. Constraint solving is an eternal challenge and a hot topic in computer science. In the past few decades, Chinese computer scientists have also made significant efforts in this area. The Boolean satisfiability (SAT) problem, the first classical NP problem, has attracted increasing attention as researchers have developed more efficient and practical algorithms for its wide applications in computer science and other fields. These algorithms can be classified into Conflict-Driven Clause Learning (CDCL)-based and local search-based, while the former is complete with low efficiency; in contrast, the latter is more efficient, but incomplete. Xishun Zhao et al. proposed an improved CDCL-based approaches by reduction based on the structure of DPLL [140, 141, 145]. Mingyu Xiao et al. improved the upper bound of SAT solving by applying parametric complexity analysis [53]. Shaowei Cai and Kaile Su significantly enhanced the efficiency of local search for SAT by examining configurations [20], a technique widely employed in various local search algorithms for the SAT problem. Meanwhile, Shaowei Cai et al. also considered applying the local search technique to solving linear integer arithmetic [19]. Also, Bican Xia investigated combining local search with cylindrical algebra decomposition to improve the efficiency of solving

SAS [150]. Meanwhile, Feifei Ma et al. considered the counting problem of SAT [94] and satisfiability modulo linear arithmetic (SMT(LA)) [95, 96], as well as optimization modulo nonlinear real arithmetic (OMT(NRA)) [129].

Nonlinear Constraint Solving and Craig Interpolation. Bican Xia, Naijun Zhan et al. considered the decidability of extensions of Tarski algebra with specific forms of exponential and trigonometric functions, as well as their combinations, and its application to the reachability analysis of hybrid and dynamical systems [42, 43, 85]. Zhilin Wu et al. gave decision procedures for several subsets of string constraints [44–46, 289].

Naijun Zhan, Bican Xia et al. provided the first algorithm for synthesizing nonlinear Craig interpolants based on Stengle's Positivstellensatz and semidefinite programming [59]. Then, they extended Motzkin's theorem to the nonlinear case, yielding an algorithm for synthesizing nonlinear Craig interpolants for quadratic concave nonlinear theories, which partially addressed the issue of shared variables in prior work [86]. Furthermore, they developed another algorithm for a nonlinear Craig interpolant for nonlinear theories subject to Archimedean conditions [87]. Finally, by homogenising unbounded optimizetion to obtain bounded optimizetion, they provided a complete solution for generating nonlinear Craig interpolants for nonlinear theories [291].

Interactive Theorem Proving. Also, in recent years, some Chinese computer scientists have focused on the implementation of (interactive) theorem provers, including Bohua Zhan's AUTO2 [356] for improving the automation of Isabelle, and HOLPY [357], which is a on-line theorem prover implemented using Python, Qinxiang Cao's VST-A for program verification [397], and Bohua Zhan et al's OSVAUTO [295] for verifying operating systems, and Mingsheng Ying et al's interactive provers based on quantum Hoare logic for quatum programs [188, 395], Shuling Wang et al's interactive theorem prover HHLProver based on Hybrid Hoare Logic (HHL) for CPS [277], as well as its automated parts HHLPy [225] and HHLPar [135].

4.1.2 Design and Verification of Safety-Critical Systems. Likewise, designing and verifying safety-critical systems remains a significant challenge in computer science and control theory.

Model-Based Design. Model-driven design methodologies, rooted in the "divide-and-conquer" principle, are the primary approach to designing complex safety-critical systems. While the industry broadly advocates for graphical modeling and simulation, which are intuitive and efficient, these approaches often lack the rigor required for safety-critical domains. Conversely, academia promotes formal methods, offering strictness but usually proving difficult to understand and inefficient in practice. Naijun Zhan and his team proposed a MARS framework [37] to address the seamless integration of graphical and formal development for safety-critical systems. It consists of a compositional graphical modeling approach by combining AADL with Simulink/Stateflow [316], alongside a formal modeling method based on Hybrid CSP (HCSP) [359], and their inter-conversion [39, 402, 403], and a refinement theory for code generation from HCSP to SystemC [325] and ANSI-C [276]. As mentioned before, researchers developed HUTP [315] to provide a unified mathematical semantic model for safety-critical systems and to ensure the theoretical correctness of model transformations. Furthermore, to formally model the mobility of CPS, they extended classic π-calculus to hybrid systems, called *hybrid π calculus* [314]. Also, Lei Bu et al. made another attempt and proposed an approach that provides scenario-based flexible modelling and scalable falsification for reconfigurable CPSs [264].

Reachable-Set Computation. Designing and verifying complex safety-critical systems relies heavily on computing reachable sets, since control synthesis, path planning, analysis, and verification naturally reduce to this task. However, researchers face the challenge that this computational problem is, in general, theoretically undecidable. Consequently, they primarily focus on approximating reachable sets; however, researchers have long faced constraints in this work due to challenges in computational efficiency, scalability, and applicability across various domains. In [85], the authors invented a decidability algorithm for an extended Tarski algebra with specific forms of exponential functions by utilizing pseudo-derivative sequences. Based on this, Gan et al.

proved that the reachability problems for three classes of linear and three classes of nonlinear dynamical systems are decidable, representing the strongest reachability results to date. Zhikun She et al. first proposed interval arithmetics-based approaches to approximate reachable sets and implemented a tool called HSolver [219]. Xue et al. proposed an efficient approximation algorithm for reachable sets based on topological homeomorphism and boundary propagation [322, 322], and an efficient approximation algorithm for reachable sets based on level sets and semidefinite programming was introduced [318]. Lei Bu et al. have invented many efficient algorithms for computing specific forms of linear dynamical and hybrid systems in bounded and unbounded time horizons [299–301, 330] and the corresponding tool, BACH [15].

Deduction Verification. Compared to model-checking based on reachable-set computation, deductive verification can perform unbounded-time verification with greater scalability. Hence, deductive verification of safety-critical systems has increasingly attracted more attention in recent years, with a powerful specification logic as the cornerstone. To this end, Naijun Zhan and his team extended Hoare logic to hybrid systems, developing Hybrid Hoare Logic (HHL), which comprises two versions: a DC-based version [187] and a generalised version without DC [360]. Both versions are mechanically supported by HHLProver [277] together with HHLPar [135] and HHLPy [225].

As with applying classic Hoare logic to program verification, the most challenging task here remains invariant generation—particularly differential invariant generation for continuous evolution, which researchers typically model with ODEs. In this respect, Chinese computer scientists made essential contributions. Currently, template-based constraint-solving techniques are a significant approach for synthesising invariants for programs and safety-critical systems. There are three challenges along this line, i.e.,

(1) conditions on a template (a predefined parametric formula) being a (differential) invariant,
(2) efficient methods to derive constraints from the template and solve the resulting constraints,
(3) how to predefine templates.

Regarding 1, Jiang Liu, Naijun Zhan, and Hengjun Zhao in [190] established a necessary and sufficient condition for parametric SAS to be a differential invariant for the considered polynomial dynamical system. Regarding point 2, theory shows—based on the necessary and sufficient condition in [190]—that template-based invariant synthesis reduces to quantifier elimination [190] or bilinear programming [48, 331]. However, quantifier elimination has double exponential complexity, and bilinear programming is NP-hard. As an alternative, Prajna and Jadbabaie introduced in their seminal work [215] a specific form of invariants called *barrier certificates* (BC). To enable efficient synthesis using semidefinite programming, the barrier-certificate condition in [215] strengthens the general condition encoding inductive invariance. Since then, researchers have devoted significant efforts to developing more relaxed (i.e., weaker) forms of barrier-certificate conditions that still admit efficient synthesis, thereby introducing, for example, exponential-type barrier certificates [144], Darboux-type barrier certificates [354], general barrier certificates [58], and vector barrier certificates [229]. To achieve efficient synthesis, these barrier-certificate conditions share a common property: convexity. Qiuye Wang et al. filled the gap between the necessary and sufficient conditions of invariants and all kinds of BC conditions. They proposed an approach based on difference of convex programming to synthesise the so-called invariant BCs from the necessary and sufficient conditions [274, 275]. To address point 3, researchers have proposed several DNN-based approaches, e.g., [375, 376, 379, 380].

Chinese computer scientists have also investigated the verification and design of cyber-physical systems with complex behaviours, including time delay and stochasticity. Time delays are unavoidable and omnipresent aspects of modern control systems, often exerting a decisive influence on system control performance and safety. However, verifying time-delay systems is exceptionally challenging because the solutions of delay differential equations depend on their execution history, moving beyond simple Markov processes. Historically, system designers have often dangerously overlooked these delays. Naijun Zhan and his collaborators did a systematic work on the verification and controller synthesis for delay dynamical and hybrid systems, including: they presented a first method for simple DDEs, grounded in Taylor models and stability analysis of discrete systems [400]; Later, for general linear DDEs, they offered a verification method using spectral theory and stability analysis of dynamic systems, which was then extended to general DDEs via linearization [76]; they also generalized existing Ordinary Differential Equation (ODE) verification techniques, such as those based on topological homeomorphism and simulation, to DDEs [36, 321, 323]. Furthermore, they introduced the concept of Time-Delay Hybrid Automata and proposed a synthesis algorithm for switch control in time-delay hybrid systems [5]. They also made initial attempts to verify stochastic dynamical and hybrid systems [75, 319, 320].

In addition, Zhenhua Duan, Cong Tian, et al. considered extending ITL with projection and proposed a so-called *propositional projection temporal logic* (PPTL) [67, 248]. Based on PPTL, Zhenhua Duan and his collaborators developed a systematic logical approach called *MSVL* [201, 327] for the design of safety-critical systems. With MSVL, developers can first specify a safety-critical system with PPTL and model it in MSVL. They can then check whether the MSVL model (program) satisfies the PPTL specification through simulation, model checking, or theorem proving. Finally, they can automatically generate correct ANSI-C code from the verified MSVL model.

4.1.3 Program Verification. Ensuring the correctness of programs is one of the fundamental scientific problems in computer science, dating back to the advent of the modern computer. As advocated by Tony Hoare [119], it becomes increasingly critical and challenging as safety-critical applications and software-defined trends emerge. In this subsection, we will summarize the efforts of Chinese computer scientists in this regard in the past decades.

Invariant Generation and Termination Analysis. The Floyd-Hoare-Naur Inductive Assertion Approach has become the dominant approach for program verification, in which termination analysis and invariant generation are the two most challenging problems.

Researchers have proposed numerous approaches to invariant generation in the literature, and they have found that constraint-solving-based program invariant synthesis is gaining traction. In this approach, they predefine a parametric invariant template and then encode the (inductive) invariant conditions into constraints. Researchers face a long-standing challenge in efficiently solving or reducing the encoded constraints, since existing approaches either require quantifier elimination with double-exponential complexity or reduce to non-convex optimizetion problems, which are generally NP-hard and lack efficient solvers. To attack the challenge, in [292], Hao Wu et al. proposed a novel algorithm for synthesizing SAS invariants of polynomial programs by exploiting Lasserre's SOS hierarchy to reduce a non-convex program (a bilinear program) to a sequence of convex programs (semidefinite programs). Furthermore, they proved that their algorithm is sound, convergent, and weakly complete under a specific robustness assumption on templates. Moreover, in recent years, neural-symbolic computation has become popular for invariant generation. E.g., Fei He et al. introduced a novel interval sample-based approach for loop invariant generation [311], and Shiwen Yu et al. proposed an approach to invariant generation by combining reinforcement learning and SMT solving [350]. Xiaoxing Ma et al. proposed the approaches to leveraging LLMs to improve the capabilities for invariant generation [21, 286]. As a generalizetion, uninterpreted predicate solving is a fundamental problem in formal verification, including loop-invariant and Constrained Horn Clauses predicate solving. Ji Wang, Shiwen Yu et al. proposed a novel discrete neural architecture combined with the Abstract Gradient Descent (AGD) algorithm to solve uninterpreted predicates in discrete hypothesis spaces directly. AGD introduces abstract gradients for discrete neurons, with computation rules defined in an abstract domain. Their tool achieves significant progress on uninterpreted predicate solving tasks [349].

The termination problem of programs is equivalent to the well-known halting problem and, hence, is undecidable in general. A complete method for termination analysis for programs, even for linear or polynomial programs, is therefore impossible. In [167], Yangjia Li et al. identified a subclass of polynomial programs in which all expressions are polynomials, and all test conditions are Boolean combinations of polynomial equations, and proved the decidability of the termination. Fei He et al. developed data-driven algorithms to prove termination and disprove non-termination (for recurrent sets) in both classical programs [102, 310] and probabilistic programs [30]. In addition, Fei He and his team proposed the first regression termination analysis framework for evolving programs [104] and developed highly efficient regression analysis algorithms for predicate abstraction [348] and interprocedural verification [108].

Static Analysis and Automatic Verification. Static analysis is essential for understanding and reasoning about the behaviours and properties of programs without executing them. Yue Li and Tian Tan at Nanjing University have been dedicated to addressing a series of fundamental and challenging research problems in program analysis, including pointer analysis (also known as points-to analysis or alias analysis) [159–161, 241–243], as well as complex language feature analysis such as reflection analysis [162–165] and native code analysis [361]. They have developed a new program analysis platform for Java called Tai-e [240] on top of which more effective foundational analysis techniques are built [149, 202]. They have also developed novel analysis methods for various software frameworks on Tai-e, including data persistence frameworks [179], microservices frameworks [368], and Android frameworks [40]. These methods have enabled the identification of real security vulnerabilities and quality issues in practical applications that rely on these frameworks, which previous work failed to detect.

Scaling sophisticated static analyses to large codebases has remained a key challenge in program analysis research for decades. Zhiqiang Zuo and his collaborators at Nanjing University proposed a systematic program analysis that scales sophisticated, context-sensitive, flow-sensitive, and even path-sensitive analyses to ten million lines of code. Their key idea is to leverage massive computing resources to accelerate the static analysis tasks. They proposed a series of studies for various analysis tasks and computing resources, including the disk-based [268], GPU-accelerated [408], and cluster-based [406] [98] systems for Context-Free-Language (CFL) reachability analysis, the disk-based [409][279] and cluster-based [237] systems for general dataflow analysis, and the system [407] for path-sensitive analysis, and have achieved promising results. To address the challenges posed by frequent code evolution, Yu Wang et al. at Nanjing University introduced an incremental approach to value-flow analysis [266]. To address the practical utility of static analysis warnings, they proposed a deep learning–based bug-prediction approach to enhance accuracy [280].

Pursuing the automatic analysis and verification of critical systems, Ji Wang and his collaborators, Wei Dong, Zhenbang Chen, Liqian Chen, and Liangze Yin, at the National University of Defence Technology, have contributed to program analysis and verification via various formal methods, including model checking, abstract interpretation, and symbolic execution. They are the earliest group to work on model checking of software systems over the model level and program level in China, such as model checking and testing Statecharts [64, 152, 263] as well as C programs [333, 334]. Following this line of research, Yin et al. presented a scheduling-constraint-based abstraction-refinement method for bounded model checking of multi-threaded C programs [338]. The notion of Event Order Graph (EOG) is proposed, along with two graph-based algorithms over EOG for counterexample validation and refinement generation, aiming to obtain a small yet effective refinement constraint. The implementation tool YOGAR-CBMC [335] won the first gold medal in the Concurrency Safety category of the International Competition on Software Verification (SV-COMP) for China in 2017 and continued to win in 2018 and 2019. They extended the method for weak memory models [336], developed parallel implementations [337], and created incremental updates [339], all of which researchers have successfully applied.

Abstract Interpretation is a general theory of sound approximation of program behaviors. It provides a powerful framework for automatically inferring properties over programs by 'executing' them on abstract domains. Through the international exchange program, Ji Wang sent Liqian Chen to ENS, and the research on abstract interpretation in China began to develop. With the collaboration of Patrick Cousot and Antoine Miné, they devised a sound floating-point polyhedra abstract domain [32] and a set of efficient yet precise abstract domains [31, 33–35]. They have designed specific abstract domains and successfully applied abstract interpretation to analysis of interrupt driven programs [293] and programs involving machine integer semantics together with bit-vector operations[70].

For symbolic verification of software, Ji Wang and Zhenbang Chen et al. have contributed to symbolic execution, including coupling path exploration with constraint solving and optimising path exploration. First, they propose to utilize the constraint solver in a white-box manner and explore the reuse possibilities of path exploration and constraint solving in tightly-coupled symbolic execution [49, 227, 228, 370]. It provides novel perspectives and approaches for addressing the challenges of constraint-solving technologies within the context of symbolic execution. Second, they presented a novel guidance method for steering path exploration in symbolic execution and a path slicing approach, both specifically targeted at regular property verification [347, 371]. By leveraging symbolic execution, they proposed a symbolic verification method for Message Passing Interface (MPI) programs [346], which is leading the static analysis tools for MPI programs. To combine abstract interpretation and symbolic execution, they have presented block-wise abstract interpretation combining abstract domains with SMT [131], and built a tool called AISE [185] which won the first golden medal for China in the Loop track of the ReachSafety category of SV-COMP 2025 [10].

Verification of Concurrent Programs. Concurrency is a ubiquitous feature in modern software systems, often employed to harness the power of multi-core processors and optimize performance. However, ensuring correctness in concurrent programs is challenging because their execution involves nondeterministic interleavings among multiple threads. Testing is limited because it often fails to locate and reproduce bugs that may appear only in a few specific program execution interleavings. Xinyu Feng and his collaborators did a systematic work on the formal verification of concurrent programs. First, they presented a rely-guarantee-based simulation technique called RGSim for verifying concurrent program refinement [175, 176]. Later, based on RGSim, they studied several applications of concurrent program refinement verification. Specifically, they developed a program logic, LiLi, for verifying the linearizability and progress properties of concurrent objects [171–173, 177, 178]. They also proposed a practical verification framework for preemptive OS kernels and applied it to verify a commercial preemptive OS μC/OS-II [304]. Additionally, they developed a framework for verifying the compilation and optimization of concurrent programs [130, 355]. They also provided methods for verifying conflict-free replicated data types in distributed systems [174] and for verifying concurrent randomized programs [72]. In addition, Fei He and his collaborators developed an ordering-consistency theory together with decision algorithms and further built a dedicated SMT solver for concurrent program verification [71, 105, 191, 236]. Specifically, they developed the verification tool Deagle [106], which won the Concurrency Safety category of SV-COMP in 2022, 2023, and 2025.

Verification of Probabilistic Programs. Probabilistic programming is a widely used paradigm for describing stochastic models as executable computer programs. Static analysis of probabilistic programs aims to algorithmically derive tight, guaranteed outcomes for stochastic properties, e.g., probabilities and expected values. Hongfei Fu et al. have made contributions in various aspects of static analysis of probabilistic programs. These contributions include termination analysis (both qualitative and quantitative) [24, 27–29, 126], expected cost analysis [25, 26, 272], probabilistic assertion analysis [235, 265], sensitivity analysis [271] and Bayesian posterior analysis [273]. Mingshuai Chen and his collaborators have contributed a taxonomy of formal techniques for reasoning about (quantitative) fixed points that capture the semantics of infinite-state loopy probabilistic programs [77, 169]. The addressed tasks include the verification (and refutation) of sound upper bounds [9], lower bounds [74], and exact semantics [38, 142, 143] (for Bayesian inference). A synthesis complements the verification perspective framework for (semi-)automatically generating quantitative loop invariants [8] as well as an efficient procedure for deciding (positive) almost-sure termination [210]. Di Wang et al. proposed various type theory-related techniques for the expected-cost analysis [61, 256, 258], static analysis [255, 259], and inference

[257] of probabilistic programs. Further contributions in this thread include Yijun Feng et al.'s result on synthesizing quantitative polynomial loop invariants [81] and Yuxin Deng et al.'s results on assertion-based logic for local reasoning [290] and termination analysis [309] for probabilistic programs.

Verification of Quantum Programs. Quantum software is indispensable for harnessing the power of quantum computers; however, program design is notoriously error-prone due to the distinctive quantum features of no-cloning and entanglement. Effective verification methods are therefore essential. Mingsheng Ying introduced the first full-fledged quantum Hoare logic, together with a novel proof of relative completeness [342]. Researchers subsequently extended this framework to cover quantum programs with classical variables [80], parallel composition [345], nondeterministic choices [79], and distributed quantum programs with classical communication [78]. To improve automation, further work reduced invariant [344] and ranking function generation [166] of quantum programs to semidefinite programming. Alternative verification approaches include projector-based quantum Hoare logic [398] and relational Hoare logic [6]. On the implementation side, Naijun Zhan's group built a theorem prover for Ying's original logic in Isabelle/HOL [189]. At the same time, Zhou et al. [396] developed a general-purpose platform for quantum programming and verification in Coq. Ying provides a comprehensive introduction to this line of research in his monograph [343].

4.1.4 Hardware Verification.

Formal verification of hardware designs refers to a family of techniques that have rigorous mathematical foundations, such as SAT solvers and model checking, to prove whether the hardware Register-Transfer Level (RTL) designs conform to their specifications, whether two hardware designs are equivalent, or whether CPU designs conform to the Instruction Set Architecture (ISA) specifications. Compared with testing and simulation, formal verification offers completeness and thus provides a higher level of safety/security assurance for hardware designs.

Several research groups at ISCAS, ECNU, and HKUST (Guangzhou) have achieved notable results in the formal verification of hardware designs, including hardware model checking, ISA compatibility checking, equivalence checking, and assertion-based verification. On hardware model checking, Qiusong Yang et al. proposed several optimization techniques for the classical IC3 algorithm (short for Incremental Construction of Inductive Clauses for Indubitable Correctness, aka, Property Directed Reachability (PDR)), including fine-tuning of SAT solvers [233], lemma prediction [232], and optimization strategies for critical proof obligations [399]. Jianwen Li et al. proposed the concept of "i-good lemmas" to improve IC3/PDR [298]. Hongce Zhang et al. proposed DeepIC3, where graph neural networks are integrated to improve the result of local inductive generalization of IC3 [122]. Cong Tian, Zhenhua Duan, and others improved the efficiency of software/hardware model checking by developing techniques to detect spurious counterexamples and refine abstract models in counterexample-guided abstraction refinement (CEGAR) [249, 250]. On formal verification of ISA-compatibility of RISC-V CPU designs, Zhilin Wu et al. developed ChiRVFormal [224], a RISC-V ISA-compliant formal verification tool for RISC-V CPU Chisel designs, which includes a parameterised Chisel reference model for the RISC-V ISA and novel mechanisms for synchronising the reference model with RISC-V CPU designs that incorporate complex microarchitecture features. On equivalence checking and assertion-based verification, Shaowei Cai et al. proposed to integrate exact simulation into sweeping for datapath combinational equivalence checking and improve the performance of the open-source equivalence checker ABC [206] for more than two orders of magnitude [51]. Hongce Zhang et al. proposed AssertLLM, a framework that uses large language models (LLMs) to automatically generate assertions from natural-language specification files (rather than sentences) [326].

4.1.5 Security Verification.

Cryptographic algorithms play a pivotal role in safeguarding sensitive information. However, their implementations are often vulnerable to various physical attacks, including power-side-channel, timing-side-channel, and fault-injection attacks. Therefore, verifying the security of cryptographic implementations against physical attacks is essential but challenging, because security properties are non-functional and their verification extends beyond traditional correctness verification. Early efforts have proposed verification techniques targeting some physical attacks, but they are limited in accuracy and scalability.

Fu Song and his collaborators did a systematic work on the security verification of cryptographic implementations. They proposed a refinement-based approach for verifying first-order power side-channel security, which synergistically integrates a fast but inaccurate semantic rule-based approach and an accurate but flow constraint solving-based approach for refining inference rules [92, 363], bringing the best of both worlds. Researchers extended the refinement-based verification approach to handle arithmetic programs in an assume-guarantee-based compositional style, thereby further improving verification efficiency and usability [90, 91]. They also generalised a refinement-based approach and a compositional reasoning approach to verify higher-order power side-channel security, using a GPU-accelerated parallel algorithm for constraint solving and an algorithm for automatically inferring assumptions [88, 89, 93]. They propose a refinement-based approach to verify timing-side-channel security, which synergistically integrates a flow- and context-sensitive lightweight taint analysis with an accurate yet flow-self-composition-based refinement of taints [18, 217]. To verify the security of cryptographic implementations against fault injection attacks, a novel SAT-encoding approach was proposed that reduces the verification problem to SAT solving, taking all potential faults into account [238]. Additionally, a first-round-based composition verification method was introduced to handle entire cryptographic implementations [238]. Moreover, researchers from Zhejiang University and their collaborators presented a generalised security-preserving refinement technique for verifying information-flow security of concurrent systems [234]. Theoretically, researchers can represent many security properties as different kinds of opacity, classifying them into state-based and action-based categories. In the untimed setting, researchers have proven that state-based and action-based opacity are decidable for finite-state automata. However, in general, both state-based and action-based opacity become undecidable in the timed setting, e.g., as represented by timed automata [22]. Naijun Zhan and his collaborators established a hierarchy of state-based and action-based opacity for subclasses of timed automata [3, 270].

4.1.6 AI Verification. With the great success of deep learning-based AI in many application domains, as advocated by Jifeng He, guaranteeing the trustworthiness of AI is becoming increasingly important and challenging [110]. Over the past decade, Chinese computer scientists have increasingly focused on this issue, primarily on verifying neural networks, deep learning systems, and black-box systems.

In the context of formal verification of neural networks, Lijun Zhang et al. proposed a symbolic-propagation-based method to enhance the precision of neural network verification via abstract interpretation [329]. Wang Lin et al. presented an approach that transforms the robustness verification problem into an equivalent nonlinear optimizetion problem [184]. Min Zhang et al. proposed provably tightest linear over-approximation techniques and acceleration methods for efficient and precise neural network verification [100, 374]. Jingyi Wang et al. established various techniques for repairing deep neural networks with provable guarantees [198, 199]. On the verification of deep learning programs, most current research focuses on properties or defects at the model level. Yingfei Xiong et al. applied static analysis based on abstract interpretation to detect numerical defects in deep learning programs using tailored abstraction techniques, such as tensor partition abstraction, that suit the characteristics of these programs [373]. On the formal verification of black-box systems, Bai Xue et al. proposed a probably approximately correct (PAC) model checking method for verifying finite-time continuous "black-box" dynamical systems [324]. Moreover, Bai Xue et al. presented a practical framework to analyze the robustness properties of deep neural networks (DNNs) [151], which first abstracts the local behaviour of a DNN via an affine model with the PAC guarantee based on black-box learning, and then infers the corresponding PAC-model robustness property. Complementary to neural network verification, Zengyu Liu et al. proposed a guess-and-check-based framework to automatically synthesise sufficient preconditions for guaranteeing safety and robustness postconditions [193]. Min Zhang et al. proposed a unified framework for the formal verification of neural network-controlled systems that combines qualitative and quantitative approaches by defining and training multiple barrier certificates [384]. They also introduced CEGAR into the development process of verifiably safe deep reinforcement learning systems, which involves iteratively training neural network controllers on abstracted and refined system states guided by counterexamples [134].

4.2 The CCF Formal Methods Technical Committee; Major FM Teams and Venues

With the rise of increasingly safety-critical applications, growing support from the government and industry, and a steadily expanding research community, the establishment of a formal-methods organizetion became both timely and necessary. Thanks to the vision of pioneers such as Chaochen Zhou, Wei Li, Jifeng He, and Huimin Lin, and the dedicated efforts of Ji Wang, Xuandong Li, Zhiming Liu, Zhenhua Duan, Naijun Zhan, and many others, the CCF Technical Committee on Formal Methods (CCF-TCFM) was formally founded in 2016. The first edition of CCF-TCFM (2016–2019) comprised approximately 120 members. Huimin Lin was the director, and Yuxi Fu, Yi Wang, and Ji Wang were the vice directors, with Naijun Zhan as the secretary. The second edition of CCF-TCFM (2020–2023) consisted of approximately 180 members. Ji Wang served as the director, with Yuxi Fu, Geguang Pu, and Naijun Zhan as vice-directors, and Wei Dong as secretary. Currently, the third edition of CCF-TCFM (2024–2026) comprises approximately 240 members. Naijun Zhan is the director, and Wei Dong, Min Zhang, and Cong Tian are the vice directors, with Zhilin Wu as the secretary.

Currently, major FM teams in China include those at ISCAS, ECNU, Nanjing University, the National University of Defense Technology, Peking University, Tsinghua University, Beihang University, Xidian University, Shanghai Jiaotong University, Tongji University, Zhejiang University, and Capital Normal University, along with industrial groups at Huawei, Ant Group, the China Aerospace Science and Technology Corporation (CASC), and others.

In addition to domestic teams, the development of formal methods in China has long benefited from the engagement of prominent foreign researchers. A notable example is Jean-Raymond Abrial—the founder of B-Method [1] and Z notation [231] for developing safety-critical software systems. Abrial served for many years as an Adjunct Professor at ECNU, where he worked closely with local researchers, including the group led by Jifeng He, contributing to both foundational advances and industrial-scale applications of formal methods. His collaboration notably supported the development of safety-critical software for metro-train control systems, including the first driverless metro in Shanghai. As a leading researcher in the Program of Intelligence Introduction for Innovation at Chinese Universities, Abrial also played an active role in bringing international academic resources to China. In recognition of these contributions, he received the International Science and Technology Cooperation Award of the People's Republic of China in 2016 [68].

The CCF-TCFM holds its annual conference, FMAC, each year in early winter. Since 2020, FMAC has been held jointly with NASAC (the National Software and Applications Conference) — the annual joint conference of CCF-TCSE (CCF Technical Committee on Software Engineering) and CCF-TCSS (CCF Technical Committee on System Software). This joint conference, known as *CCF ChinaSoft*, has grown into the most influential academic venue for software theories, tools, and applications in China, as evidenced by its 2024 attendance of more than 2,500.

Chinese computer scientists also initiated several international conferences to promote academic exchange and strengthen collaboration between the Chinese and the global FM communities. The first was TASE (International Symposium on Theoretical Aspects of Software Engineering), launched by Jifeng He in 2006; Another is SETTA (Symposium on Dependable Software Engineering Theories, Tools, and Applications), which was established in 2015 through the efforts of Chinese researchers—including Chaochen Zhou, Zhenhua Duan, Zhiming Liu, Ji Wang, Xuandong Li, Naijun Zhan, and others—together with international colleagues such as Cliff Jones, Deepak Kapur, Martin Fränzle, Kim Larsen, Sriram Rajamani, and Kwangkeun Yi.

In addition to the China-initiated conferences, many FM-related international conferences have been held in China, including CONFESTA (the joint conference comprising CONCUR, FORMATS, QEST, and SETTA) in 2018, FM (International Symposium on Formal Methods) in 2021, as well as multiple editions of ATVA (International Symposium on Automated Technology for Verification and Analysis), ICFEM (International Conference on Formal Engineering Methods), MEMOCODE (ACM-IEEE International Symposium on Formal Methods and Models for System Design), and ICECCS (International Conference on Engineering of Complex Computer Systems).

5 Industry Contributions and Applications

This section provides a brief overview of the industrial contributions and applications of formal methods in China. It primarily involves safety-critical domains such as aerospace, transportation, telecommunications, chip design, operating systems, and finance. In many of these domains, software assurance standards—such as DO-178B/C, DO-333, CC EAL 1–7, and SIL 1–4—explicitly mandate or encourage the adoption of formal methods as a means to achieve high assurance levels.

Aerospace. Supported by the NSFC's Grand Research Program on Fundamentals of Trustworthy Software (2007–2016), China has established a comprehensive framework of trustworthy assurance techniques for aerospace embedded software development, leveraging formal methods such as model checking, abstract interpretation, and deductive verification; see details in [114]. Representative contributions and applications include: (i) A joint research and engineering effort between the China Academy of Space Technology (CAST) and ISCAS, which developed the MARS toolchain [37] for modeling, analysis, and verification of hybrid systems. MARS was later applied to the verification of control programs for the Chang'e-3 lunar lander [378] and the Tianwen-1 Mars spacecraft; (ii) The work by the research team at the National University of Defense Technology, which proposed sequentialization-based static analysis for detecting numeric-related runtime errors in interrupt-driven programs [293]—a characteristic feature in aerospace software systems; (iii) The work by Yongwang Zhao et al. from Beihang University, which identified multiple errors in the widely adopted avionics embedded software standard ARINC 653 [362, 383].

Railway. Naijun Zhan's research group at ISCAS has integrated their interactive theorem prover HHL Prover [277] with the MARS toolchain [37] and applied it, in collaboration with Beijing Jiaotong University, to certify the safety of the Chinese Train Control System (CTCS) under combined scenarios [401]. The research team at Nanjing University developed the BACH toolkit [15] to verify the bounded reachability of linear hybrid automata, supported it with online compositional verification, and deployed it on the hardware-in-the-loop simulation platform of the National Engineering Research Center of Train Control Systems. Fei He et al. from Tsinghua University, developed the VCS tool for model checking component-based systems [107] and created the temporal verification framework VeRV for vehicle bus systems [367]; researchers have applied both results to analyze and verify high-speed railway train control software.

System Software. Formal methods have been extensively used in China to establish correctness guarantees for system software, e.g., *operating systems* (OS), *compilers*, and *databases*. Xinyu Feng et al. developed a practical verification framework for preemptive OS kernels and applied it to verify key modules in the commercial embedded real-time OS μC/OS-II [304]. Building on this work, Naijun Zhan et al. incorporated worst-case execution time (WCET) analysis and verified another real-time OS widely used in Chinese space missions, thereby achieving the Common Criteria Evaluation Assurance Level (CC-EAL) 5+ certification. Also, an adapted framework based on the Verified Software Toolchain (VST) [397] was applied to the verification of a variant of the distributed operating system HarmonyOS, called LiteOS-M (around 8K LoC, which also helped it pass CC-EAL 5+ certification. In parallel, Yongwang Zhao et al. proposed a similar rely-guarantee verification framework [221, 222, 382], which they used to verify ARINC 653 [362]—the de facto standard of partitioning operating systems—as well as other microkernels [312]. Yi Li and colleagues at Huawei Inc. applied a broad suite of formal methods to verify Harmony-OS kernel, leading to its certification with CC-EAL 6.[2] Moreover, Haibo Chen et al. have applied verification and optimization techniques to certify the correctness of concurrent file systems [404, 405], weak memory models [211], and database query rewrite rules [284]. Research groups at Shanghai Jiao Tong University have long pursued deductive verification of system software. For example, Qinxiang Cao et al. worked on separation-logic-based C program verification [294, 397], and Yuting Wang et al. produced verified C, Java, and Rust compilers [281, 282, 366, 372].

Beyond the surveyed applications, a substantial body of work explores the use of formal methods in industrial control systems and robotics, e.g., [60, 269, 381], as well as in the modeling and verification of security/network protocols, e.g., [99, 170, 204].

We note that the work surveyed in this section is far from being exhaustive, considering especially the ever-growing application of formal methods in Chinese technology companies—such as Huawei and ZTE in *telecommunications*, HiSilicon and Empyrean in *chip design*, NIO in *autonomous driving*, as well as Ant Group in *finance*—to strengthen their software and/or hardware reliability.

6 Challenges and Future Directions

This section outlines several key challenges and promising future directions in research, application, and talent cultivation related to formal methods. Notably, many of these considerations extend beyond China and are relevant to the global FM community.

6.1 Research and Industrial Applications

Automation and Scalability. Fully automated verification of real-world, large-scale software and hardware systems against highly expressive properties is arguably the holy grail of formal methods. While significant advancements have been made in specialized domains such as type systems for certifying type safety [213], symbolic execution for detecting bugs [139], model checking for verifying bounded reachability [11, 55], abstract interpretation for certifying runtime error-freeness [57], and theorem proving for establishing functional correctness [2], these techniques either provide *limited formal guarantee* or exhibit *poor scalability* due to the well-known state/path-explosion problem or the prerequisites of manual efforts in discovering formal specifications—such as loop invariants, pre-/post-conditions, and function contracts—that faithfully characterize intended program behaviors; See [82, Fig. 3] for a taxonomy of formal methods as per their strength of guarantee and scalability. Over the past decades, various techniques have been proposed to address this challenge, including interpolation [205], k-induction [223], IC3/PDR [14], CEGAR [56], counterexample-guided inductive synthesis (CEGIS) [8, 230], and proof-carrying code [208]. However, *specification synthesis* remains a central bottleneck in achieving full automation: Many of these methods either rely heavily on user-supplied annotations (e.g., templates) or confine push-button techniques to semi-decision or approximate procedures.

The emergence of large language models has opened new avenues for addressing the challenge mentioned above, thanks to their remarkable capabilities for contextual (code-level) understanding and reasoning. Recent studies [200, 214, 285, 287, 288]—many of which were initiated by the Chinese FM community—have investigated the use of LLMs for *automated specification synthesis*, achieving substantial improvements over conventional approaches. Nevertheless, these methods exhibit significant *scalability limitations* when applied to large-scale programs, primarily due to two factors: First, the *context-length limitation* of LLMs [73] impedes their capacity to perform holistic reasoning over monolithic codebases within a single inference step; Second, verifying complex programs typically requires synthesizing *a diverse and interdependent set of specifications*. We envisage that a modular, fine-grained framework for generating and refining formal specifications could offer a promising

[2]See the official announcement at https://www.huawei.com/cn/news/2023/8/cybersecurity-hongmengkernel-cceal6.

direction for the automated verification of large-scale programs; see, e.g., our preliminary research effort in [283], which enables the almost-fully automated verification of real-world C programs with thousands of lines of code.

Uncertainty. Our universe is inherently stochastic, uncertain, and even chaotic. Real-world computing systems can hardly, if ever, be deployed in a fixed, closed, and perfectly predictable environment. This limitation is particularly pronounced in cyber(-human)-physical systems [146, 147], which typically exhibit both (i) *exogenous uncertainty*, arising from their operation in open, dynamic environments that are inherently unpredictable, difficult to characterize fully, and often marked by discrepancies between design-time requirements/assumptions and runtime realities—in fact, the environment per se can be highly volatile or even (partially) unknown; and (ii) *endogenous uncertainty*, induced by the ever-increasing integration of data-driven, learning-enabled components that exhibit limited predictability and interpretability, such as neural network controllers and language models. As a consequence, specifying, analyzing, and verifying safety-critical cyber-human-physical systems subject to substantial uncertainty—thereby ensuring their safety, reliability, and effectiveness—remains a significant challenge in computer science, necessitating new theoretical foundations and engineering methodologies.

The challenge posed by uncertainty in system design and verification has triggered a surge of interest among the global FM community. Flagship international conferences in formal methods, programming languages, and cyber-physical systems—such as FM, CAV, OOPSLA, POPL, PLDI, and those in the CPS-IoT (Cyber-Physical Systems and Internet-of-Things) Week series—have witnessed an increasing number of workshops and sessions dedicated to uncertainty-centric topics, including probabilistic programming, stochastic systems, and quantitative verification. Chinese researchers have played a visible and impactful role in advancing this frontier. Notable contributions include Yuxin Deng's research on probabilistic process calculi [63], Lijun Zhang's results on probabilistic model checking [101], Naijun Zhan's work on stochastic hybrid systems [75], Hongfei Fu's research on quantitative termination [27], Di Wang's work on cost analysis of probabilistic programs [256], and Mingshuai Chen's results on quantitative fixed-point reasoning [169]. Importantly, this line of research also exemplifies the tight, sustained connection between Chinese FM researchers and the global FM community, as they have achieved many of these outcomes through international collaborations.

FM Meets AI. As AI technologies rapidly encroach on safety-critical infrastructures, the intersection of formal methods and AI has emerged as a pivotal research frontier, particularly in the context of autonomous systems, adaptive control, IoT-enabled infrastructures, and smart cities. This frontier encompasses two major branches: (i) *FM for AI (FM4AI)*, namely the application of formal methods and tools to specify, verify, and assure AI components or AI-powered systems. As mentioned in Section 4, the Chinese FM community has made substantial contributions in this thread towards trustworthy AI [148]. In recent years, however, the advent of large language models based on the transformer architecture [254] has brought new challenges and opportunities. Their unprecedented multimodal understanding, reasoning, and generative capabilities, along with emerging risks such as hallucinations [125] and privacy leakage [124], have triggered an intense debate about "whether, how, and to what extent we should verify LLMs via formal methods" [127]; (ii) *AI for FM (AI4FM)*, namely the use of AI techniques to advance the scalability, automation, and usability of formal methods. In addition to the aforementioned results on LLM-based specification synthesis, the Chinese FM community has also contributed various approaches to synthesizing core FM artifacts such as invariants [311, 350], variants [158, 244], interpolants [41, 183], barrier certificates [377, 379], and controllers [380], leveraging traditional AI techniques, such as decision trees, support vector machines, neural networks, and reinforcement learning. Moving forward, researchers expect the *synergistic integration of FM and AI* (see, e.g., [369]) to reshape both fields, fostering the (semi-)rigorous assurance of increasingly intelligent systems while endowing FM with unparalleled levels of automation, adaptability, and real-world applicability.

Emerging Computing Paradigms. As Moore's Law approaches its physical limits [247], the drive to transcend conventional computational boundaries has given rise to novel paradigms, including quantum, biological, neuromorphic, and photonic computing. These advances challenge the foundations of computer science, as programming languages and development methodologies rooted in the Turing machine and finite automata theory are no longer directly applicable. New abstractions and toolchains are therefore required, together with robust approaches to ensuring the correctness and reliability of safety-critical systems. For instance, the *intrinsic uncertainty* of quantum computing renders program verification considerably more difficult than in classical settings (cf. the above-mentioned challenge of uncertainty). We believe that the aforementioned rich body of research results achieved by the Chinese FM community on the analysis and verification of quantum and probabilistic programs provides important theoretical underpinnings for these emerging paradigms. Furthermore, developing effective modeling and logical frameworks for reasoning about *heterogeneous and hybrid computing* across diverse physical substrates remains a central challenge—one that calls for deep interdisciplinary collaboration among the communities of theoretical computer science, formal methods, and computer architecture.

6.2 Education and Talent Cultivation

Education plays a pivotal role in the sustained advancement of formal methods. Yet, their steep learning curve—primarily driven by the heavy reliance on mathematics and logic, as well as the limited usability and scalability of existing tools—has hindered their adoption in software development. While the ACM and IEEE curricula for computer science and software engineering both include program correctness [136, 137], surveys on formal methods education in China further highlight the need to strengthen their presence in specialized training [186, 303]. Since researchers can regard programs as formal specifications amenable to mechanized processing, educators find it both natural and essential to embed formal concepts into foundational courses (e.g., programming, data structures, and compiler construction) and to emphasize their connections to mainstream approaches in advanced courses (e.g., discrete mathematics, algorithms, software engineering, and artificial intelligence) to promote adoption and enhance proficiency [267].

A key driver of progress in China has been the cultivation of talent through both domestic and international training initiatives. Over the past two decades, FM education and training in China have gradually expanded from a small number of elite research groups to a broader network of universities and research institutes, producing a growing cohort of scholars with expertise in both theory and practice. In particular, graduate education supported by, e.g., UNU-IIST and multiple training schools, has had a lasting impact. As documented in contemporaneous reports [13], UNU-IIST in Macau served for many years as a world-class research and training center in formal methods, offering year-long fellowships to graduate students and young lecturers from developing countries, including Mainland China, providing intensive courses in specification, refinement, and mathematically based software design, followed by supervised research projects. These fellows returned to their home institutions equipped to introduce FM courses, supervise students, and seed new research groups, thereby strengthening the domestic educational infrastructure. Complementing this immersive model, Summer School on Trustworthy Software initiated by Jifeng He in ECNU, Summer School on Formal Methods initiated by Lijun Zhang and Naijun Zhan in ISCAS, and the International School on Engineering Trustworthy Software Systems (SETSS) initiated by Zhiming Liu in Southwest University, etc. provided a series of advanced training in rigorous modeling, verification, and trustworthy systems engineering for graduate students and young scholars, with the help of international researchers. These training schools educated numerous graduate students nationwide, promoting the dissemination of state-of-the-art tools and methodologies while fostering early engagement with international FM research culture.

Notably, the Grand Research Program on Fundamentals of Trustworthy Software, funded by the NSFC from 2007 to 2016, has nurtured a generation of leading scholars at the forefront of global scientific research and a strong core of mid-career researchers. However, challenges remain: the domestic FM talent pipeline is still limited in scale, the distribution of high-quality teaching resources is uneven, and opportunities for cross-disciplinary training—linking FM with areas such as AI, cybersecurity, and emerging computing paradigms—are relatively scarce. Strengthening international exchange programs, establishing joint training bases between academia and industry, and integrating FM into interdisciplinary curricula will be critical for developing a new generation of researchers and practitioners. Such initiatives will not only enhance China's capacity for original FM research and accelerate the translation of FM advances into industrial innovation but also promote global collaboration, enabling China to actively contribute to and benefit from the worldwide FM community and the development of trustworthy computing technologies.

7 Conclusion

The development of formal methods in China has followed a trajectory that is closely intertwined with, yet in several respects distinctive from, the broader international evolution of the field. FM in China began with the early establishment of mathematical logic and theoretical computer science in the 1950s, advanced through significant international engagement during the 1980s and 1990s, and expanded rapidly in the 2000s in response to both scientific opportunity and national demand for trustworthy software. Compared with global trends, three characteristics stand out:

First, the role of international exchange has been central to China's FM history. The return of scholars trained abroad in the period of reform and opening-up and the establishment of UNU/IIST in Macau served as major accelerators, creating channels through which theoretical advances—especially in semantics, refinement, real-time reasoning, and algebraic approaches—were quickly disseminated and extended. While similar patterns can be observed elsewhere, UNU/IIST's sustained influence on talent cultivation and research direction gives China's FM landscape a distinctive institutional contour.

Second, China has developed characteristic strengths in areas that combine mathematical foundations with system-level reasoning. Notable examples include the Duration Calculus and its subsequent developments, the Unifying Theories of Programming and their extensions, the mechanization of mathematics and nonlinear constraint solving, as well as the modelling and verification of real-time, hybrid systems. In these areas, Chinese researchers have produced contributions that are now well integrated into the international FM corpus. At the same time, other directions—such as SAT/SMT solving, concurrent system verification, static analysis, and the verification of probabilistic, quantum, and AI-based systems—have brought China closer into alignment with global research priorities, reflecting a worldwide convergence of theoretical and practical challenges.

Third, China has placed strong emphasis on the industrial application of formal methods to safety-critical domains, including aerospace, rail transportation, operating systems, telecommunications, and finance. The scale and national centrality of these applications—in particular, verified control software for space missions, certified train-control components, and high-assurance operating systems—position China among the few countries where FM has been systematically deployed in large, mission-critical industrial projects. This pragmatic orientation has further contributed to the growth of domestic research teams and to the formation of coordinated community structures.

Looking forward, many of the challenges facing the Chinese FM community mirror those of the global community: enhancing automation and scalability, integrating data-driven and model-based reasoning, extending verification to increasingly autonomous and stochastic systems, and strengthening interdisciplinary education and tooling. Nevertheless, the historical trajectory outlined in this article suggests that China is well-positioned to make significant contributions to the next phase of FM research and application. With a mature academic ecosystem, growing international engagement, and expanding industrial demand, China's FM community is likely to remain an active and increasingly influential participant in shaping the global development of formal methods. One potential avenue is to increase active participation in international FM standardization efforts and in global FM communities, such as Formal Methods Europe (FME), thereby contributing to the development of widely accepted global FM practices. China could also take the lead in (i) organizing and/or (co-)chairing prominent international events in formal methods and related areas, as exemplified by its involvement in CONFESTA 2018, LICS 2020, FM 2021, ICFEM 2025, QEST+FORMATS 2026, TACAS 2027, and others; and (ii) spearheading multi-national FM initiatives, particularly in emerging fields such as AI verification, quantum computing, and cybersecurity. These efforts will not only elevate China's influence on the global FM stage but also foster knowledge exchange and best practices, ensuring the continued growth and relevance of formal methods in addressing complex challenges across industries.

Acknowledgments

We appreciate Professors Huimin Lin, Jian Lu, Hong Mei, Zhenjiang Hu, Zhiming Liu, Zhenhua Duan, Xuandong Li, and Xiaoxing Ma for their comments and criticisms on the draft of this paper. We give special thanks to Professors Shaowei Cai, Hongfei Fu, Xinyu Feng, Yuan Feng, Fei He, Min Zhang, and Zhilin Wu for providing related materials on their own research. We thank Zhongyi Wang and Tengjie Lin for their assistance in formatting the extensive list of references. Moreover, we thank the anonymous reviewers for their constructive feedback.

In addition, this work has been partially funded by the National Key R&D Program of China under grant No. 2022YFA1005100 and 2022YFA1005101, by the NSFC under grant No. 62192732, 62402479, 62032024, and 62572427, by the CAS Project for Young Scientists in Basic Research under grant No. YSBR-040, by the CAS President's International Fellowship under grant No. 2024PG0016, and by the ZJNSF Major Program under grant No. LD24F020013.

References

[1] J-R Abrial, Matthew KO Lee, DS Neilson, PN Scharbach, and Ib Holm Sørensen. 1991. The B-method. In *International Symposium of VDM Europe*. Springer, 398–405. doi:10.1007/BFb0020001

[2] Wolfgang Ahrendt, Bernhard Beckert, Richard Bubel, Reiner Hähnle, Peter H. Schmitt, and Mattias Ulbrich (Eds.). 2016. *Deductive Software Verification - The KeY Book - From Theory to Practice*. Lecture Notes in Computer Science, Vol. 10001. Springer. doi:10.1007/978-3-319-49812-6

[3] Jie An, Qiang Gao, Lingtai Wang, Naijun Zhan, and Ichiro Hasuo. 2024. The Opacity of Timed Automata. In *FM 2024 (Lecture Notes in Computer Science, Vol. 14933)*. Springer, 620–637. doi:10.1007/978-3-031-71162-6_32

[4] Jie An, Naijun Zhan, Xiaoshan Li, Miaomiao Zhang, and Wang Yi. 2018. Model Checking Bounded Continuous-time Extended Linear Duration Invariants. In *HSCC 2018*. ACM, 81–90. doi:10.1145/3178126.3178147

[5] Yunjun Bai, Ting Gan, Li Jiao, Bican Xia, Bai Xue, and Naijun Zhan. 2021. Switching controller synthesis for delay hybrid systems under perturbations. In *HSCC 2021*. ACM, 3:1–3:11. doi:10.1145/3447928.3456657

[6] Gilles Barthe, Justin Hsu, Mingsheng Ying, Nengkun Yu, and Li Zhou. 2020. Relational proofs for quantum programs. *Proc. ACM Program. Lang.* 4, POPL (2020), 21:1–21:29. doi:10.1145/3371089

[7] Rana Barua, Suman Roy, and Chaochen Zhou. 1999. Completeness of Neighbourhood Logic. In *STACS 1999 (Lecture Notes in Computer Science, Vol. 1563)*. Springer, 521–530. doi:10.1007/3-540-49116-3_49

[8] Kevin Batz, Mingshuai Chen, Sebastian Junges, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. 2023. Probabilistic Program Verification via Inductive Synthesis of Inductive Invariants. In *TACAS 2023 (2) (Lecture Notes in Computer Science, Vol. 13994)*. Springer, 410–429. doi:10.1007/978-3-031-30820-8_25

[9] Kevin Batz, Mingshuai Chen, Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Philipp Schröer. 2021. Latticed k-Induction with an Application to Probabilistic Programs. In *CAV 2021 (2) (Lecture Notes in Computer Science, Vol. 12760)*. Springer, 524–549. doi:10.1007/978-3-030-81688-9_25

[10] Dirk Beyer and Jan Strejček. 2025. Improvements in software verification and witness validation: SV-COMP 2025. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 151–186. doi:10.1007/978-3-031-90660-2_9

[11] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. 1999. Symbolic Model Checking without BDDs. In *TACAS 1999 (Lecture Notes in Computer Science, Vol. 1579)*. Springer, 193–207. doi:10.1007/3-540-49059-0_14

[12] Dines Bjørner, C.A.R. Hoare, Jonathan P. Bowen, Jifeng He, Hans Langmaack, Ernst-Rüdiger Olderog, Ursula H. Martin, Victoria Stavridou, Fleming Nielson, Hanne Riis Nielson, Howard Barringer, Douglas Edwards, Hans H. Løvengreen, Anders P. Ravn, and Hans Rischel. 1989. A ProCoS Project Description: ESPRIT BRA 3104. *Bulletin of the European Association of Theoretical Computer Science* 39 (1989), 60–73. https://api.semanticscholar.org/CorpusID:117144235

[13] Jonathan P. Bowen. 1999. Beijing takes the software pearl in Macau's crown. https://www.timeshighereducation.com/news/beijing-takes-the-software-pearl-in-macaus-crown/148725.article. The Times Higher Education Supplement 1409, 13. Retrieved November 27, 2025.

[14] Aaron R. Bradley. 2011. SAT-Based Model Checking without Unrolling. In *VMCAI 2011 (Lecture Notes in Computer Science, Vol. 6538)*. Springer, 70–87. doi:10.1007/978-3-642-18275-4_7

[15] Lei Bu, You Li, Linzhang Wang, and Xuandong Li. 2008. BACH: Bounded ReAchability CHecker for Linear Hybrid Automata. In *FMCAD 2008*. IEEE, 1–4. doi:10.1109/FMCAD.2008.ECP.13

[16] Bruno Buchberger. 1976. Some properties of Gröbner-bases for polynomial ideals. *SIGSAM Bull.* 10, 4 (1976), 19–24. doi:10.1145/1088222.1088224

[17] Ricky W. Butler. 2001. What is Formal Methods? https://shemesh.larc.nasa.gov/fm/fm-what.html. Retrieved July 17, 2025.

[18] Luwei Cai, Fu Song, and Taolue Chen. 2024. Towards Efficient Verification of Constant-Time Cryptographic Implementations. *Proc. ACM Softw. Eng.* 1, FSE (2024), 1019–1042. doi:10.1145/3643772

[19] Shaowei Cai, Bohan Li, and Xindi Zhang. 2023. Local Search For Satisfiability Modulo Integer Arithmetic Theories. *ACM Trans. Comput. Log.* 24, 4 (2023), 32:1–32:26. doi:10.1145/3597495

[20] Shaowei Cai, Kaile Su, and Abdul Sattar. 2011. Local search with edge weighting and configuration checking heuristics for minimum vertex cover. *Artif. Intell.* 175, 9–10 (2011), 1672–1696. doi:10.1016/j.artint.2011.03.003

[21] Weining Cao, Guangyuan Wu, Tangzhi Xu, Yuan Yao, Hengfeng Wei, Taolue Chen, and Xiaoxing Ma. 2025. Clause2Inv: A Generate-Combine-Check Framework for Loop Invariant Inference. *Proc. ACM Softw. Eng.* 2, ISSTA (2025), 1009–1030. doi:10.1145/3728920

[22] Franck Cassez. 2009. The Dark Side of Timed Opacity. In *ISA 2009 (Lecture Notes in Computer Science, Vol. 5576)*. Springer, 21–30. doi:10.1007/978-3-642-02617-1_3

[23] CCF. 2011. *Introduction to Xu Jiafu*. Lifetime Achievement Award. CCF Rewards. China Computer Federation. https://www.ccf.org.cn/Awards/Awards_Recipients/2011/Lifetime_Contribution/2011-02-16/578028.shtml.

[24] Krishnendu Chatterjee, Hongfei Fu, and Amir Kafshdar Goharshady. 2016. Termination Analysis of Probabilistic Programs Through Positivstellensatz's. In *CAV 2016 (Lecture Notes in Computer Science, Vol. 9779)*. Springer, 3–22. doi:10.1007/978-3-319-41528-4_1

[25] Krishnendu Chatterjee, Hongfei Fu, Amir Kafshdar Goharshady, and Nastaran Okati. 2018. Computational Approaches for Stochastic Shortest Path on Succinct MDPs. In *IJCAI 2018*. 4700–4707. doi:10.24963/ijcai.2018/653

[26] Krishnendu Chatterjee, Hongfei Fu, and Aniket Murhekar. 2017. Automated Recurrence Analysis for Almost-Linear Expected-Runtime Bounds. In *CAV 2017 (Lecture Notes in Computer Science, Vol. 10426)*. Springer, 118–139. doi:10.1007/978-3-319-63387-9_6

[27] Krishnendu Chatterjee, Hongfei Fu, and Petr Novotný. 2020. Termination Analysis of Probabilistic Programs with Martingales. In *Foundations of Probabilistic Programming*, Alexandra Silva, Gilles Barthe, and Joost-Pieter Katoen (Eds.). Cambridge University Press, 221–258. doi:10.1017/9781108770750

[28] Krishnendu Chatterjee, Hongfei Fu, Petr Novotný, and Rouzbeh Hasheminezhad. 2016. Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. In *POPL 2016*. ACM, 327–342. doi:10.1145/3174800

[29] Krishnendu Chatterjee, Hongfei Fu, Petr Novotný, and Rouzbeh Hasheminezhad. 2018. Algorithmic Analysis of Qualitative and Quantitative Termination Problems for Affine Probabilistic Programs. *ACM Trans. Program. Lang. Syst.* 40, 2 (2018), 7:1–7:45. doi:10.1145/3174800

[30] Jianhui Chen and Fei He. 2020. Proving almost-sure termination by omega-regular decomposition. In *PLDI 2020*. ACM, 869–882. doi:10.1145/3385412.3386002

[31] Liqian Chen, Jiangchao Liu, Antoine Miné, Deepak Kapur, and Ji Wang. 2014. An Abstract Domain to Infer Octagonal Constraints with Absolute Value. In *SAS 2014 (Lecture Notes in Computer Science, Vol. 8723)*. Springer, 101–117. doi:10.1007/978-3-319-10936-7_7

[32] Liqian Chen, Antoine Miné, and Patrick Cousot. 2008. A Sound Floating-Point Polyhedra Abstract Domain. In *APLAS 2008 (Lecture Notes in Computer Science, Vol. 5356)*. Springer, 3–18. doi:10.1007/978-3-540-89330-1_2

[33] Liqian Chen, Antoine Miné, Ji Wang, and Patrick Cousot. 2009. Interval Polyhedra: An Abstract Domain to Infer Interval Linear Relationships. In *SAS 2009 (Lecture Notes in Computer Science, Vol. 5673)*. Springer, 309–325. doi:10.1007/978-3-642-03237-0_21

[34] Liqian Chen, Antoine Miné, Ji Wang, and Patrick Cousot. 2010. An Abstract Domain to Discover Interval Linear Equalities. In *VMCAI 2010 (Lecture Notes in Computer Science, Vol. 5944)*. Springer, 112–128. doi:10.1007/978-3-642-11319-2_11

[35] Liqian Chen, Antoine Miné, Ji Wang, and Patrick Cousot. 2011. Linear Absolute Value Relation Analysis. In *ESOP 2011 (Lecture Notes in Computer Science, Vol. 6602)*. Springer, 156–175. doi:10.1007/978-3-642-19718-5_9

[36] Mingshuai Chen, Martin Fränzle, Yangjia Li, Peter Nazier Mosaad, and Naijun Zhan. 2016. Validated Simulation-Based Verification of Delayed Differential Dynamics. In *FM 2016 (Lecture Notes in Computer Science, Vol. 9995)*. 137–154. doi:10.1007/978-3-319-48989-6_9

[37] Mingshuai Chen, Xiao Han, Tao Tang, Shuling Wang, Mengfei Yang, Naijun Zhan, Hengjun Zhao, and Liang Zou. 2017. MARS: A Toolchain for Modelling, Analysis and Verification of Hybrid Systems. In *Provably Correct Systems*. Springer, 39–58. doi:10.1007/978-3-319-48628-4_3

[38] Mingshuai Chen, Joost-Pieter Katoen, Lutz Klinkenberg, and Tobias Winkler. 2022. Does a Program Yield the Right Distribution? Verifying Probabilistic Programs via Generating Functions. In *CAV 2022 (Lecture Notes in Computer Science, Vol. 13371)*. Springer, 79–101.

doi:10.1007/978-3-031-13185-1_5
[39] Mingshuai Chen, Anders P. Ravn, Shuling Wang, Mengfei Yang, and Naijun Zhan. 2016. A Two-Way Path Between Formal and Informal Design of Embedded Systems. In *UTP 2016 (Lecture Notes in Computer Science, Vol. 10134)*. Springer, 65–92. doi:10.1007/978-3-319-52228-9_4
[40] Menglong Chen, Tian Tan, Minxue Pan, and Yue Li. 2025. PacDroid: A Pointer-Analysis-Centric Framework for Security Vulnerabilities in Android Apps. In *ICSE 2025*. IEEE, 2803–2815. doi:10.1109/ICSE55347.2025.00208
[41] Mingshuai Chen, Jian Wang, Jie An, Bohua Zhan, Deepak Kapur, and Naijun Zhan. 2019. NIL: Learning Nonlinear Interpolants. In *CADE 2019 (Lecture Notes in Computer Science, Vol. 11716)*. Springer, 178–196. doi:10.1007/978-3-030-29436-6_11
[42] Rizeng Chen, Haokun Li, Bican Xia, Tianqi Zhao, and Tao Zheng. 2024. Isolating all the real roots of a mixed trigonometric-polynomial. *J. Symb. Comput.* 121 (2024), 102250. doi:10.1016/j.jsc.2023.102250
[43] Rizeng Chen and Bican Xia. 2023. Deciding first-order formulas involving univariate mixed trigonometric-polynomials. In *ISSAC 2023*. ACM, 145–154. doi:10.1145/3597066.3597104
[44] Taolue Chen, Yan Chen, Matthew Hague, Anthony W. Lin, and Zhilin Wu. 2018. What is decidable about string constraints with the ReplaceAll function. *Proc. ACM Program. Lang.* 2, POPL (2018), 3:1–3:29. doi:10.1145/3158091
[45] Taolue Chen, Alejandro Flores-Lamas, Matthew Hague, Zhilei Han, Denghang Hu, Shuanglong Kan, Anthony W. Lin, Philipp Rümmer, and Zhilin Wu. 2022. Solving string constraints with Regex-dependent functions through transducers with priorities and variables. *Proc. ACM Program. Lang.* 6, POPL (2022), 1–31. doi:10.1145/3498707
[46] Taolue Chen, Matthew Hague, Anthony W. Lin, Philipp Rümmer, and Zhilin Wu. 2019. Decision procedures for path feasibility of string-manipulating programs with complex operations. *Proc. ACM Program. Lang.* 3, POPL (2019), 49:1–49:30. doi:10.1145/3290362
[47] Xin Chen, Jifeng He, Zhiming Liu, and Naijun Zhan. 2007. A Model of Component-Based Programming. In *FSEN 2007 (Lecture Notes in Computer Science, Vol. 4767)*. Springer, 191–206. doi:10.1007/978-3-540-75698-9_13
[48] Xin Chen, Chao Peng, Wang Lin, Zhengfeng Yang, Yifang Zhang, and Xuandong Li. 2020. A Novel Approach for Solving the BMI Problem in Barrier Certificates Generation. In *CAV 2020 (Lecture Notes in Computer Science, Vol. 12224)*. Springer, 582–603. doi:10.1007/978-3-030-53288-8_29
[49] Zhenbang Chen, Zehua Chen, Ziqi Shuai, Guofeng Zhang, Weiyu Pan, Yufeng Zhang, and Ji Wang. 2021. Synthesize solving strategy for symbolic execution. In *ISSTA 2021*. ACM, 348–360. doi:10.1145/3460319.3464815
[50] Zhenbang Chen, Zhiming Liu, Anders P. Ravn, Volker Stolz, and Naijun Zhan. 2009. Refinement and verification in component-based model-driven design. *Sci. Comput. Program.* 74, 4 (2009), 168–196. doi:10.1016/j.scico.2008.08.003
[51] Zhihan Chen, Xindi Zhang, Yuhang Qian, Qiang Xu, and Shaowei Cai. 2023. Integrating Exact Simulation into Sweeping for Datapath Combinational Equivalence Checking. In *ICCAD 2023*. IEEE, 1–9. doi:10.1109/ICCAD57390.2023.10323876
[52] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. 1994. *Machine Proofs in Geometry: Automated Production of Readable Proofs for Geometry Theorems*. World Scientific. doi:10.1142/2196 Series on Applied Mathematics, Vol 6.
[53] Huairui Chu, Mingyu Xiao, and Zhe Zhang. 2021. An improved upper bound for SAT. *Theor. Comput. Sci.* 887 (2021), 51–62. doi:10.1016/j.tcs.2021.06.045
[54] Alonzo Church. 1936. An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics* 58, 2 (1936), 345–363. doi:10.2307/2371045
[55] Edmund M. Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. 2001. Bounded Model Checking Using Satisfiability Solving. *Formal Methods Syst. Des.* 19, 1 (2001), 7–34. doi:10.1023/A:1011276507260
[56] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. 2000. Counterexample-Guided Abstraction Refinement. In *CAV 2000 (Lecture Notes in Computer Science, Vol. 1855)*. Springer, 154–169. doi:10.1007/10722167_15
[57] Patrick Cousot and Radhia Cousot. 1977. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *POPL 1977*. ACM, 238–252. doi:10.1145/512950.512973
[58] Liyun Dai, Ting Gan, Bican Xia, and Naijun Zhan. 2017. Barrier certificates revisited. *J. Symb. Comput.* 80 (2017), 62–86. doi:10.1016/j.jsc.2016.07.010
[59] Liyun Dai, Bican Xia, and Naijun Zhan. 2013. Generating Non-linear Interpolants by Semidefinite Programming. In *CAV 2013 (Lecture Notes in Computer Science, Vol. 8044)*. Springer, 364–380. doi:10.1007/978-3-642-39799-8_25
[60] Wenbin Dai and Valeriy Vyatkin. 2012. Redesign distributed PLC control systems using IEC 61499 function blocks. *IEEE Transactions on Automation Science and Engineering* 9, 2 (2012), 390–401. doi:10.1109/TASE.2012.2188794
[61] Ankush Das, Di Wang, and Jan Hoffmann. 2023. Probabilistic Resource-Aware Session Types. *Proc. ACM Program. Lang.* 7, POPL (2023), 1925–1956. doi:10.1145/3571259
[62] Jim Davies, Tomasz Janowski, Adegboyega K. Ojo, and Aadya Shukla. 2007. Technological foundations of electronic governance. In *ICEGOV 2007 (ACM International Conference Proceeding Series, Vol. 232)*. ACM, 5–11. doi:10.1145/1328057.1328063
[63] Yuxin Deng. 2015. *Semantics of Probabilistic Processes: An Operational Approach*. Springer. doi:10.1007/978-3-662-45198-4
[64] Wei Dong, Ji Wang, Xuan Qi, and Zhichang Qi. 2001. Model Checking UML Statecharts. In *(APSEC 2001*. IEEE Computer Society, 363–370. doi:10.1109/APSEC.2001.991503

[65] Yunmei Dong. 2002. Recursive functions of context free languages (I) - The definitions of CFPRF and CFRF. *Sci. China Ser. F Inf. Sci.* 45, 1 (2002), 25–39. doi:10.1360/02yf9002

[66] Yunmei Dong. 2002. Recursive functions of context free languages (II) - Validity of CFPRF and CFRF definitions. *Sci. China Ser. F Inf. Sci.* 45, 2 (2002), 81–102. doi:10.1360/02yf9007

[67] Zhenhua Duan, Nan Zhang, and Maciej Koutny. 2013. A complete proof system for propositional projection temporal logic. *Theor. Comput. Sci.* 497 (2013), 84–107. doi:10.1016/j.tcs.2012.01.026

[68] ECNU News. 2017. ECNU foreign expert Jean-Raymond Abrial Won the International Science and Technology Cooperation Award. https://sei.ecnu.edu.cn/seien/87/2b/c33262a362283/page.htm. Retrieved November 27, 2025.

[69] Marcel Erné, Jürgen Koslowski, Austin Melton, and George E Strecker. 1993. A primer on Galois connections. *Annals of the New York Academy of Sciences* 704, 1 (1993), 103–125. doi:10.1111/j.1749-6632.1993.tb52513.x

[70] Guangsheng Fan, Liqian Chen, Banghu Yin, Wenyu Zhang, Peisen Yao, and Ji Wang. 2025. Program Analysis Combining Generalized Bit-Level and Word-Level Abstractions. *Proc. ACM Softw. Eng.* 2, ISSTA (2025), 663–685. doi:10.1145/3728905

[71] Hongyu Fan, Weiting Liu, and Fei He. 2022. Interference relation-guided SMT solving for multi-threaded program verification. In *PPoPP 2022*. ACM, 163–176. doi:10.1145/3503221.3508424

[72] Weijie Fan, Hongjin Liang, Xinyu Feng, and Hanru Jiang. 2025. A Program Logic for Concurrent Randomized Programs in the Oblivious Adversary Model. In *ESOP 2025 (Lecture Notes in Computer Science, Vol. 15694)*. Springer, 322–348. doi:10.1007/978-3-031-91118-7_13

[73] Chongzhou Fang, Ning Miao, Shaurya Srivastav, Jialin Liu, Ruoyu Zhang, Ruijie Fang, Asmita, Ryan Tsang, Najmeh Nazari, Han Wang, and Houman Homayoun. 2024. Large Language Models for Code Analysis: Do LLMs Really Do Their Job?. In *USENIX Security Symposium 2024*. USENIX Association. https://www.usenix.org/system/files/usenixsecurity24-fang.pdf

[74] Shenghua Feng, Mingshuai Chen, Han Su, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Naijun Zhan. 2023. Lower Bounds for Possibly Divergent Probabilistic Programs. *Proc. ACM Program. Lang.* 7, OOPSLA1 (2023), 696–726. doi:10.1145/3586051

[75] Shenghua Feng, Mingshuai Chen, Bai Xue, Sriram Sankaranarayanan, and Naijun Zhan. 2020. Unbounded-Time Safety Verification of Stochastic Differential Dynamics. In *CAV 2020 (2) (Lecture Notes in Computer Science, Vol. 12225)*. Springer, 327–348. doi:10.1007/978-3-030-53291-8_18

[76] Shenghua Feng, Mingshuai Chen, Naijun Zhan, Martin Fränzle, and Bai Xue. 2019. Taming Delays in Dynamical Systems - Unbounded Verification of Delay Differential Equations. In *CAV 2019 (Lecture Notes in Computer Science, Vol. 11561)*. Springer, 650–669. doi:10.1007/978-3-030-25540-4_37

[77] Shenghua Feng, Tengshun Yang, Mingshuai Chen, and Naijun Zhan. 2024. A Unified Framework for Quantitative Analysis of Probabilistic Programs. In *Principles of Verification (1) (Lecture Notes in Computer Science, Vol. 15260)*. Springer, 230–254. doi:10.1007/978-3-031-75783-9_10

[78] Yuan Feng, Sanjiang Li, and Mingsheng Ying. 2022. Verification of Distributed Quantum Programs. *ACM Trans. Comput. Log.* 23, 3 (2022), 19:1–19:40. doi:10.1145/3517145

[79] Yuan Feng and Yingte Xu. 2023. Verification of Nondeterministic Quantum Programs. In *ASPLOS 2023 (3)*. ACM, 789–805. doi:10.1145/3582016.3582039

[80] Yuan Feng and Mingsheng Ying. 2021. Quantum Hoare logic with classical variables. *ACM Trans. Quantum Comput.* 2, 4 (2021), 1–43. doi:10.1145/3456877

[81] Yijun Feng, Lijun Zhang, David N. Jansen, Naijun Zhan, and Bican Xia. 2017. Finding Polynomial Loop Invariants for Probabilistic Programs. In *ATVA 2017 (Lecture Notes in Computer Science, Vol. 10482)*. Springer, 400–416. doi:10.1007/978-3-319-68167-2_26

[82] Kathleen Fisher, John Launchbury, and Raymond Richards. 2017. The HACMS program: using formal methods to eliminate exploitable bugs. *Philos. Trans. R. Soc.* 375, 2104 (2017), 20150401. doi:10.1098/rsta.2015.0401

[83] The State Council Investigation Team for the "7·23" Especially Serious Accident on the Yongwen Railway Line. 2011. *Investigation Report on the Especially Serious "7·23" Accident on the Yongwen Railway Line.* Technical Report. State Administration of Work Safety (China). https://www.chinanews.com/gn/2011/12-28/3567137.shtml

[84] Yuxi Fu. 2003. Bisimulation congruence of chi calculus. *Inf. Comput.* 184, 1 (2003), 201–226. doi:10.1016/S0890-5401(03)00061-0

[85] Ting Gan, Mingshuai Chen, Yangjia Li, Bican Xia, and Naijun Zhan. 2018. Reachability Analysis for Solvable Dynamical Systems. *IEEE Trans. Autom. Control.* 63, 7 (2018), 2003–2018. doi:10.1109/TAC.2017.2763785

[86] Ting Gan, Liyun Dai, Bican Xia, Naijun Zhan, Deepak Kapur, and Mingshuai Chen. 2016. Interpolant Synthesis for Quadratic Polynomial Inequalities and Combination with EUF. In *IJCAR 2016 (Lecture Notes in Computer Science, Vol. 9706)*. Springer, 195–212. doi:10.1007/978-3-319-40229-1_14

[87] Ting Gan, Bican Xia, Bai Xue, Naijun Zhan, and Liyun Dai. 2020. Nonlinear Craig Interpolant Generation. In *CAV 2020 (Lecture Notes in Computer Science, Vol. 12224)*. Springer, 415–438. doi:10.1007/978-3-030-53288-8_20

[88] Pengfei Gao, Fu Song, and Taolue Chen. 2024. Compositional Verification of First-Order Masking Countermeasures against Power Side-Channel Attacks. *ACM Trans. Softw. Eng. Methodol.* 33, 3 (2024), 79:1–79:38. doi:10.1145/3635707

[89] Pengfei Gao, Hongyi Xie, Fu Song, and Taolue Chen. 2021. A Hybrid Approach to Formal Verification of Higher-Order Masked Arithmetic Programs. *ACM Trans. Softw. Eng. Methodol.* 30, 3 (2021), 26:1–26:42. doi:10.1145/3428015

[90] Pengfei Gao, Hongyi Xie, Pu Sun, Jun Zhang, Fu Song, and Taolue Chen. 2022. Formal Verification of Masking Countermeasures for Arithmetic Programs. *IEEE Trans. Software Eng.* 48, 3 (2022), 973–1000. doi:10.1109/TSE.2020.3008852

[91] Pengfei Gao, Hongyi Xie, Jun Zhang, Fu Song, and Taolue Chen. 2019. Quantitative Verification of Masked Arithmetic Programs Against Side-Channel Attacks. In *TACAS 2019*, Vol. 11427. Springer, 155–173. doi:10.1007/978-3-030-17462-0_9

[92] Pengfei Gao, Jun Zhang, Fu Song, and Chao Wang. 2019. Verifying and Quantifying Side-channel Resistance of Masked Software Implementations. *ACM Trans. Softw. Eng. Methodol.* 28, 3 (2019), 16:1–16:32. doi:10.1145/3330392

[93] Pengfei Gao, Yedi Zhang, Fu Song, Taolue Chen, and François-Xavier Standaert. 2023. Compositional Verification of Efficient Masking Countermeasures against Side-Channel Attacks. *Proc. ACM Program. Lang.* 7, OOPSLA2 (2023), 1817–1847. doi:10.1145/3622862

[94] Cunjing Ge, Feifei Ma, Tian Liu, Jian Zhang, and Xutong Ma. 2018. A New Probabilistic Algorithm for Approximate Model Counting. In *IJCAR 2018 (Lecture Notes in Computer Science, Vol. 10900)*. Springer, 312–328. doi:10.1007/978-3-319-94205-6_21

[95] Cunjing Ge, Feifei Ma, Xutong Ma, Fan Zhang, Pei Huang, and Jian Zhang. 2019. Approximating Integer Solution Counting via Space Quantification for Linear Constraints. In *IJCAI 2019*. 1697–1703. https://dl.acm.org/doi/abs/10.5555/3367243.3367274

[96] Cunjing Ge, Feifei Ma, Peng Zhang, and Jian Zhang. 2018. Computing and estimating the volume of the solution space of SMT(LA) constraints. *Theor. Comput. Sci.* 743 (2018), 110–129. doi:10.1016/j.tcs.2016.10.019

[97] Kurt Gödel. 1967. On Formally Undecidable Propositions of *Principia Mathematica* and Related Systems I. In *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Harvard University Press, Cambridge, MA, 596–616. https://archive.org/details/fromfregetogodel0025unse/page/664/mode/2up Originally published in *Monatshefte für Mathematik und Physik*, 38:173–198 (1931).

[98] Rong Gu, Zhiqiang Zuo, Xi Jiang, Han Yin, Zhaokang Wang, Linzhang Wang, Xuandong Li, and Yihua Huang. 2021. Towards Efficient Large-Scale Interprocedural Program Static Analysis on Distributed Data-Parallel Computation. *IEEE Trans. Parallel Distributed Syst.* 32, 4 (2021), 867–883. doi:10.1109/IPDPS.2019.00086

[99] Jingjing Guan, Hui Li, XiangDong Li, XiaoLei Wang, Binghan Wang, Qiuye Wang, Shengchao Qin, Mengda He, Md Armanuzzaman, and Ziming Zhao. 2025. Formally Verifying the State Machine of TLS 1.3 Handshake in OpenSSL. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 1–10. doi:10.1109/INFOCOM55648.2025.11044576

[100] Xingwu Guo, Wenjie Wan, Zhaodi Zhang, Min Zhang, Fu Song, and Xuejun Wen. 2021. Eager falsification for accelerating robustness verification of deep neural networks. In *ISSRE 2021*. IEEE, 345–356. doi:10.1109/ISSRE52982.2021.00044

[101] Ernst Moritz Hahn, Yi Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. 2014. IscasMc: A Web-Based Probabilistic Model Checker. In *FM 2014 (Lecture Notes in Computer Science, Vol. 8442)*. Springer, 312–317. doi:10.1007/978-3-319-06410-9_22

[102] Zhilei Han and Fei He. 2023. Data-driven Recurrent Set Learning For Non-termination Analysis. In *ICSE 2023*. IEEE, 1303–1315. doi:10.1109/ICSE48619.2023.00115

[103] Michael R. Hansen and Chaochen Zhou. 1991. Semantics and Completeness of Duration Calculus. In *Real-Time: Theory in Practice, REX Workshop (Lecture Notes in Computer Science, Vol. 600)*. Springer, 209–225. doi:10.1007/BFb0031994

[104] Fei He and Jitao Han. 2020. Termination analysis for evolving programs: An incremental approach by reusing certified modules. *Proc. ACM Program. Lang.* 4, OOPSLA (2020), 199:1–199:27. doi:10.1145/3428267

[105] Fei He, Zhihang Sun, and Hongyu Fan. 2021. Satisfiability modulo ordering consistency theory for multi-threaded program verification. In *PLDI 2021*. ACM, 1264–1279. doi:10.1145/3453483.3454108

[106] Fei He, Zhihang Sun, and Hongyu Fan. 2022. Deagle: An SMT-based Verifier for Multi-threaded Programs (Competition Contribution). In *TACAS 2022 (Lecture Notes in Computer Science, Vol. 13244)*. Springer, 424–428. doi:10.1007/978-3-030-99527-0_25

[107] Fei He, Liangze Yin, Bow-Yaw Wang, Lianyi Zhang, Guanyu Mu, and Wenrui Meng. 2013. VCS: A Verifier for Component-Based Systems. In *ATVA 2013 (Lecture Notes in Computer Science, Vol. 8172)*. Springer, 478–481. doi:10.1007/978-3-319-02444-8_39

[108] Fei He, Qianshan Yu, and Liming Cai. 2022. Efficient Summary Reuse for Software Regression Verification. *IEEE Trans. Software Eng.* 48, 4 (2022), 1417–1431. doi:10.1109/TSE.2020.3021477

[109] Jifeng He. 2002. Integrating CSP and DC. In *ICECCS 2002*. IEEE Computer Society, 47. doi:10.1109/ICECCS.2002.1181497

[110] Jifeng He. 2019. Safe and Trustworthy Artificial Intelligence. *J. Inf. Secur. Commun. Secr.* 10 (2019), 4–8.

[111] Jifeng He and C. A. R. Hoare. 1999. Linking Theories in Probabilistic Programming. *Inf. Sci.* 119, 3–4 (1999), 205–218. doi:10.1016/S0020-0255(99)00015-8

[112] Jifeng He, C. A. R. Hoare, Martin Fränzle, Markus Müller-Olm, Ernst-Rüdiger Olderog, Michael Schenke, Michael R. Hansen, Anders P. Ravn, and Hans Rischel. 1994. Provably Correct Systems. In *FTRTFT 1994 (Lecture Notes in Computer Science, Vol. 863)*. Springer, 288–335. doi:10.1007/3-540-58468-4_171

[113] Jifeng He, Xiaoshan Li, and Zhiming Liu. 2006. rCOS: A refinement calculus of object systems. *Theor. Comput. Sci.* 365, 1–2 (2006), 109–142. doi:10.1016/j.tcs.2006.07.034

[114] Jifeng He, Zhiguang Shan, Ji Wang, Geguang Pu, Yufei Fang, Ke Liu, Ruizhen Zhao, and Zhaotian Zhang. 2018. Review of the Achievements of Major Research Plan on "Trustworthy Software". *Science Foundation in China* 32, 3 (2018), 291–296. (In Chinese), https://www.nsfc.gov.cn/csc/20345/20348/pdf/2018/201803291.pdf.

[115] Jifeng He and Huibiao Zhu. 2000. Formalising VERILOG. In *ICECS 2000*. IEEE, 412–415. doi:10.1109/ICECS.2000.911568

[116] Matthew Hennessy and Huimin Lin. 1993. Proof Systems for Message-Passing Process Algebras. In *CONCUR 1993 (Lecture Notes in Computer Science, Vol. 715)*. Springer, 202–216. doi:10.1007/3-540-57208-2_15

[117] Matthew Hennessy and Huimin Lin. 1995. Symbolic Bisimulations. *Theor. Comput. Sci.* 138, 2 (1995), 353–389. doi:10.1016/0304-3975(94)00172-F

[118] David Hilbert. 1998. The Grounding of Elementary Number Theory. In *From Brouwer to Hilbert: The Debate on the Foundations of Mathematics in the 1920s*. Oxford University Press, New York, 266–273. https://global.oup.com/ushe/product/from-brouwer-to-hilbert-9780195096323?cc=cn&lang=en&facet_narrowbypubdate_facet=Older&facet_narrowbypubdate_facet=Older

[119] C. A. R. Hoare. 2003. The verifying compiler: A grand challenge for computing research. *J. ACM* 50, 1 (2003), 63–69. doi:10.1007/978-3-540-30579-8_5

[120] C. A. R. Hoare and J. He. 1998. *Unifying Theories of Programming*. Prentice Hall, Englewood Cliffs. https://www.cs.ox.ac.uk/publications/publication8324-abstract.html

[121] Jiawei Hong. 1986. Proving by Example and Gap Theorems. In *SFCS 1986*. IEEE Computer Society, 107–116. doi:10.1109/SFCS.1986.48

[122] Guangyu Hu, Jianheng Tang, Changyuan Yu, Wei Zhang, and Hongce Zhang. 2024. DeepIC3: Guiding IC3 Algorithms by Graph Neural Network Clause Prediction. In *ASPDAC 2024*. IEEE, 262–268. doi:10.1109/ASP-DAC58780.2024.10473807

[123] Shihua Hu. 2008. *The Collection of Shihua Hu's Essays*. Academic Press.

[124] Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. 2022. Are Large Pre-Trained Language Models Leaking Your Personal Information?. In *EMNLP (Findings) 2022*. Association for Computational Linguistics, 2038–2047. https://aclanthology.org/2022.findings-emnlp.148.pdf

[125] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Trans. Inf. Syst.* 43, 2 (2025), 42:1–42:55. doi:10.1145/3703155

[126] Mingzhang Huang, Hongfei Fu, Krishnendu Chatterjee, and Amir Kafshdar Goharshady. 2019. Modular verification for almost-sure termination of probabilistic programs. *Proc. ACM Program. Lang.* 3, OOPSLA (2019), 129:1–129:29. doi:10.1145/3360555

[127] Huawei Workshop on Formal Methods. 2024. Formal Methods in the Age of Intelligence. Chaspark. https://www.chaspark.com/#/live/1030301090375839744?anchorV=1039354325153751040&multi=zh&lang=en. Retrieved August 13, 2025.

[128] Dang Van Hung and Chaochen Zhou. 1999. Probabilistic Duration Calculus for Continuous Time. *Formal Aspects Comput.* 11, 1 (1999), 21–44. doi:10.1007/s001650050034

[129] Fuqi Jia, Yuhang Dong, Rui Han, Pei Huang, Minghao Liu, Feifei Ma, and Jian Zhang. 2025. A Complete Algorithm for Optimization Modulo Nonlinear Real Arithmetic. In *AAAI 2025*. AAAI Press, 11255–11263. doi:10.1609/aaai.v39i11.33224

[130] Hanru Jiang, Hongjin Liang, Siyang Xiao, Junpeng Zha, and Xinyu Feng. 2019. Towards certified separate compilation for concurrent programs. In *PLDI 2019*. ACM, 111–125. doi:10.1145/3314221.3314595

[131] Jiahong Jiang, Liqian Chen, Xueguang Wu, and Ji Wang. 2017. Block-Wise Abstract Interpretation by Combining Abstract Domains with SMT. In *VMCAI 2017 (Lecture Notes in Computer Science, Vol. 10145)*. Springer, 310–329. doi:10.1007/978-3-319-52234-0_17

[132] XinJie Jiang and YongSen Xu. 1988. NUSL: An executable specification language based on data abstraction. In *Proceedings of the 2nd VDM-Europe Symposium on VDM—The Way Ahead*. Springer-Verlag, Berlin, Heidelberg, 124–138. doi:10.1007/3-540-50214-9_12

[133] Xinjie Jiang and Yongsen Xu. 1990. Diverse executable semantics definitions in NUSL and an implementation of functional types. *SIGPLAN Not.* 25, 5 (1990), 39––52. doi:10.1145/382080.382631

[134] Peng Jin, Jiaxu Tian, Dapeng Zhi, Xuejun Wen, and Min Zhang. 2022. Trainify: A CEGAR-driven training and verification framework for safe deep reinforcement learning. In *CAV 2022*. Springer, 193–218. doi:10.1007/978-3-031-13185-1_10

[135] Xiangyu Jin, Bohua Zhan, Shuling Wang, and Naijun Zhan. 2024. HHLPar: Automated Theorem Prover for Parallel Hybrid Communicating Sequential Processes. *CoRR* abs/2407.08936 (2024). doi:10.48550/arXiv.2407.08936

[136] Joint Task Force on Computing Curricula (ACM and IEEE). 2013. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. (2013), 27–38. doi:10.1145/2534860

[137] Joint Task Force on Computing Curricula (ACM and IEEE). 2014. Software Engineering 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. (2014), 10–19. https://dl.acm.org/doi/book/10.1145/2965631

[138] Wei Ke, Xiaoshan Li, Zhiming Liu, and Volker Stolz. 2012. rCOS: A formal model-driven engineering method for component-based software. *Frontiers Comput. Sci. China* 6, 1 (2012), 17–39. doi:10.1007/s11704-012-2901-5

[139] James C. King. 1976. Symbolic Execution and Program Testing. *Commun. ACM* 19, 7 (1976), 385–394. doi:10.1145/360248.360252

[140] Hans Kleine Büning, K. Subramani, and Xishun Zhao. 2007. Boolean Functions as Models for Quantified Boolean Formulas. *J. Autom. Reason.* 39, 1 (2007), 49–75. doi:10.1007/s10817-007-9067-0

[141] Hans Kleine Büning and Xishun Zhao. 2004. Equivalence Models for Quantified Boolean Formulas. In *SAT 2004 (Lecture Notes in Computer Science, Vol. 3542)*. Springer, 224–234. doi:10.1007/11527695_18

[142] Lutz Klinkenberg, Christian Blumenthal, Mingshuai Chen, Darion Haase, and Joost-Pieter Katoen. 2024. Exact Bayesian Inference for Loopy Probabilistic Programs using Generating Functions. *Proc. ACM Program. Lang.* 8, OOPSLA1 (2024), 923–953. doi:10.1145/3649844

[143] Lutz Klinkenberg, Tobias Winkler, Mingshuai Chen, and Joost-Pieter Katoen. 2023. Exact Probabilistic Inference Using Generating Functions. In *LAFI 2023*. [Extended Abstract], https://popl23.sigplan.org/details/lafi-2023-papers/5/Exact-Probabilistic-Inference-Using-Generating-Functions.

[144] Hui Kong, Fei He, Xiaoyu Song, William N. N. Hung, and Ming Gu. 2013. Exponential-condition-based barrier certificate generation for safety verification of hybrid systems. In *CAV 2013 (Lecture Notes in Computer Science, Vol. 8044)*. Springer, 242–257. doi:10.1007/978-3-642-39799-8_17

[145] Oliver Kullmann and Xishun Zhao. 2013. On Davis-Putnam reductions for minimally unsatisfiable clause-sets. *Theor. Comput. Sci.* 492 (2013), 70–87. doi:10.1016/j.tcs.2013.04.020

[146] Edward A. Lee. 2015. The Past, Present and Future of Cyber-Physical Systems: A Focus on Models. *Sensors* 15, 3 (2015), 4837–4869. doi:10.3390/s150304837

[147] Edward A. Lee. 2016. Fundamental Limits of Cyber-Physical Systems Modeling. *ACM Trans. Cyber Phys. Syst.* 1, 1 (2016), 3:1–3:26. doi:10.1145/2912149

[148] Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. 2023. Trustworthy AI: From Principles to Practices. *ACM Comput. Surv.* 55, 9 (2023), 177:1–177:46. doi:10.1145/3555803

[149] Chenxi Li, Haoran Lin, Tian Tan, and Yue Li. 2025. Two Approaches to Fast Bytecode Frontend for Static Analysis. *Proc. ACM Program. Lang.* OOPSLA (2025). doi:10.1145/3763081

[150] Haokun Li, Bican Xia, and Tianqi Zhao. 2023. Local Search for Solving Satisfiability of Polynomial Formulas. In *CAV 2023 (Lecture Notes in Computer Science, Vol. 13965)*. Springer, 87–109. doi:10.1007/978-3-031-37703-7_5

[151] Renjue Li, Pengfei Yang, Cheng-Chao Huang, Youcheng Sun, Bai Xue, and Lijun Zhang. 2022. Towards Practical Robustness Analysis for DNNs based on PAC-Model Learning. In *ICSE 2022*. ACM, 2189–2201. doi:10.1145/3510003.3510143

[152] Shuhao Li, Ji Wang, and Zhi-Chang Qi. 2004. Property-Oriented Test Generation from UML Statecharts. In *ASE 2004*. IEEE Computer Society, 122–131. doi:10.1109/ASE.2004.1342730

[153] Wei Li. 1983. *An operational approach to semantics and translation for programming languages*. Ph. D. Dissertation. University of Edinburgh, UK. https://era.ed.ac.uk/handle/1842/6636

[154] Wei Li. 1990. A Type-Theoretic Approach to Program Development. *J. Comput. Sci. Technol.* 5, 3 (1990), 209–224. doi:10.1007/BF02945309

[155] Wei Li. 1993. An open logic system. *Science in China Series A - Mathematics, Physics, Astronomy & Technological Science* 36, 3 (1993), 362–375.

[156] Wei Li and Yuefei Sui. 2022. *R-Calculus, II: Many-Valued Logics*. Springer. doi:10.1007/978-981-16-9294-9

[157] Xuandong Li and Dang Van Hung. 1996. Checking Linear Duration Invariants by Linear Programming. In *ASIAN 1996 (Lecture Notes in Computer Science, Vol. 1179)*. Springer, 321–332. doi:10.1007/BFb0027804

[158] Yi Li, Xuechao Sun, Yong Li, Andrea Turrini, and Lijun Zhang. 2019. Synthesizing Nested Ranking Functions for Loop Programs via SVM. In *ICFEM 2019 (Lecture Notes in Computer Science, Vol. 11852)*. Springer, 438–454. doi:10.1007/978-3-030-32409-4_27

[159] Yue Li, Tian Tan, Anders Møller, and Yannis Smaragdakis. 2018. Precision-guided context sensitivity for pointer analysis. *Proc. ACM Program. Lang.* 2, OOPSLA (2018), 141:1–141:29. doi:10.1145/3276511

[160] Yue Li, Tian Tan, Anders Møller, and Yannis Smaragdakis. 2018. Scalability-first pointer analysis with self-tuning context-sensitivity. In *FSE 2018*. ACM, 129–140. doi:10.1145/3236024.3236041

[161] Yue Li, Tian Tan, Anders Møller, and Yannis Smaragdakis. 2020. A Principled Approach to Selective Context Sensitivity for Pointer Analysis. *ACM Trans. Program. Lang. Syst.* 42, 2 (2020), 10:1–10:40. doi:10.1145/3381915

[162] Yue Li, Tian Tan, Yulei Sui, and Jingling Xue. 2014. Self-inferencing Reflection Resolution for Java. In *ECOOP 2014 (Lecture Notes in Computer Science, Vol. 8586)*. Springer, 27–53. doi:10.1007/978-3-662-44202-9_2

[163] Yue Li, Tian Tan, and Jingling Xue. 2015. Effective Soundness-Guided Reflection Analysis. In *SAS 2015 (Lecture Notes in Computer Science, Vol. 9291)*. Springer, 162–180. doi:10.1007/978-3-662-48288-9_10

[164] Yue Li, Tian Tan, and Jingling Xue. 2019. Understanding and Analyzing Java Reflection. *ACM Trans. Softw. Eng. Methodol.* 28, 2 (2019), 7:1–7:50. doi:10.1145/3295739

[165] Yue Li, Tian Tan, Yifei Zhang, and Jingling Xue. 2016. Program Tailoring: Slicing by Sequential Criteria. In *ECOOP 2016 (LIPIcs, Vol. 56)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 15:1–15:27. doi:10.4230/LIPIcs.ECOOP.2016.15

[166] Yangjia Li and Mingsheng Ying. 2018. Algorithmic analysis of termination problems for quantum programs. *Proc. ACM Program. Lang.* 2, POPL (2018), 35:1–35:29. doi:10.1145/3158123

[167] Yangjia Li, Naijun Zhan, Mingshuai Chen, Hui Lu, Guohua Wu, and Joost-Pieter Katoen. 2015. On Termination of Polynomial Programs with Equality Conditions. *CoRR* abs/1510.05201 (2015). doi:10.48550/arXiv.1510.05201

[168] Zhoujun Li and Huowang Chen. 1998. Checking Strong/Weak Bisimulation Equivalences and Observation Congruence for the pi-Calculus. In *ICALP 1998 (Lecture Notes in Computer Science, Vol. 1443)*. Springer, 707–718. doi:10.1007/BFb0055095

[169] Zhiyang Li, Mingqi Yang, Shenghua Feng, and Mingshuai Chen. 2025. Fixed-Point Reasoning for Stochastic Systems: A Survey of Recent Advancements and Open Challenges. In *Design and Verification of Cyber-Physical Systems: From Theory to Applications*. Springer. [To appear].

[170] Zechun Li, Peng Zhang, Yichi Zhang, and Hongkun Yang. 2025. NDD: A Decision Diagram for Network Verification. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. 237–258. https://www.usenix.org/conference/nsdi25/presentation/li-zechun

[171] Hongjin Liang and Xinyu Feng. 2013. Modular verification of linearizability with non-fixed linearization points. In *PLDI 2013*. ACM, 459–470. doi:10.1145/2491956.2462189

[172] Hongjin Liang and Xinyu Feng. 2016. A program logic for concurrent objects under fair scheduling. In *POPL 2016*. ACM, 385–399. doi:10.1145/2837614.2837635

[173] Hongjin Liang and Xinyu Feng. 2018. Progress of concurrent objects with partial methods. *Proc. ACM Program. Lang.* 2, POPL 2018 (2018), 20:1–20:31. doi:10.1145/3158108

[174] Hongjin Liang and Xinyu Feng. 2021. Abstraction for conflict-free replicated data types. In *PLDI 2021*. ACM, 636–650. doi:10.1145/3453483.3454067

[175] Hongjin Liang, Xinyu Feng, and Ming Fu. 2012. A rely-guarantee-based simulation for verifying concurrent program transformations. In *POPL 2012*. ACM, 455–468. doi:10.1145/2103656.2103711

[176] Hongjin Liang, Xinyu Feng, and Ming Fu. 2014. Rely-Guarantee-Based Simulation for Compositional Verification of Concurrent Program Transformations. *ACM Trans. Program. Lang. Syst.* 36, 1 (2014), 3:1–3:55. doi:10.1145/2576235

[177] Hongjin Liang, Xinyu Feng, and Zhong Shao. 2014. Compositional verification of termination-preserving refinement of concurrent programs. In *CSL-LICS 2014*. ACM, 65:1–65:10. doi:10.1145/2603088.2603123

[178] Hongjin Liang, Jan Hoffmann, Xinyu Feng, and Zhong Shao. 2013. Characterizing Progress Properties of Concurrent Objects via Contextual Refinements. In *CONCUR 2013 (Lecture Notes in Computer Science, Vol. 8052)*. Springer, 227–241. doi:10.1007/978-3-642-40184-8_17

[179] Yufei Liang, Teng Zhang, Ganlin Li, Tian Tan, Chang Xu, Chun Cao, Xiaoxing Ma, and Yue Li. 2025. Pointer Analysis for Database-Backed Applications. *Proc. ACM Program. Lang.* 9, PLDI (2025), 1417–1441. doi:10.1145/3729307

[180] Huimin Lin. 1991. PAM: A Process Algebra Manipulator. In *CAV 1991 (Lecture Notes in Computer Science, Vol. 575)*. Springer, 136–146. doi:10.1007/3-540-55179-4_14

[181] Huimin Lin. 1995. PAM: A Process Algebra Manipulator. *Formal Methods Syst. Des.* 7, 3 (1995), 243–259. doi:10.1007/bf01384078

[182] Huimin Lin. 1998. Complete Proof Systems for Observation Congruences in Finite-Control pi-Calculus. In *ICALP 1998 (Lecture Notes in Computer Science, Vol. 1443)*. Springer, 443–454. doi:10.1007/BFb0055074

[183] Wang Lin, Mi Ding, Kaipeng Lin, and Zuohua Ding. 2023. Formal synthesis of neural Craig interpolant via counterexample guided deep learning. *Inf. Softw. Technol.* 163 (2023), 107298. doi:10.1016/j.infsof.2023.107298

[184] Wang Lin, Zhengfeng Yang, Xin Chen, Qingye Zhao, Xiangkun Li, Zhiming Liu, and Jifeng He. 2019. Robustness Verification of Classification Deep Neural Networks via Linear Programming. In *CVPR 2019*. Computer Vision Foundation / IEEE, 11418–11427. doi:10.1109/CVPR.2019.01168

[185] Yao Lin, Zhenbang Chen, and Ji Wang. 2025. AISE v2.0: Combining Loop Transformations - (Competition Contribution). In *TACAS 2025 (Lecture Notes in Computer Science, Vol. 15698)*. Springer, 199–204. doi:10.1007/978-3-031-90660-2_12

[186] Bo Liu, Zhiming Liu, Zongyan Qiu, and Xiao Qin. 2018. On computer science education of undergraduate students to improve their understanding of program correctness and to develop their skills in developing correct programs. *Computer Education* (2018). (in Chinese).

[187] Jiang Liu, Jidong Lv, Zhao Quan, Naijun Zhan, Hengjun Zhao, Chaochen Zhou, and Liang Zou. 2010. A Calculus for Hybrid CSP. In *APLAS 2010 (Lecture Notes in Computer Science, Vol. 6461)*. Springer, 1–15. doi:10.1007/978-3-642-17164-2_1

[188] Junyi Liu, Bohua Zhan, Shuling Wang, Shenggang Ying, Tao Liu, Yangjia Li, Mingsheng Ying, and Naijun Zhan. 2019. Formal Verification of Quantum Algorithms Using Quantum Hoare Logic. In *CAV 2019 (Lecture Notes in Computer Science, Vol. 11562)*. Springer, 187–207. doi:10.1007/978-3-030-25543-5_12

[189] Junyi Liu, Bohua Zhan, Shuling Wang, Shenggang Ying, Tao Liu, Yangjia Li, Mingsheng Ying, and Naijun Zhan. 2019. Formal Verification of Quantum Algorithms Using Quantum Hoare Logic. In *CAV 2019 (2) (Lecture Notes in Computer Science, Vol. 11562)*. Springer, 187–207. doi:10.1007/978-3-030-25543-5_12

[190] Jiang Liu, Naijun Zhan, and Hengjun Zhao. 2011. Computing semi-algebraic invariants for polynomial dynamical systems. In *EMSOFT 2011*. ACM, 97–106. doi:10.1145/2038642.2038659

[191] Jiangyi Liu, Fengmin Zhu, and Fei He. 2023. Automated Ambiguity Detection in Layout-Sensitive Grammars. *Proc. ACM Program. Lang.* 7, OOPSLA2 (2023), 1150–1175. doi:10.1145/3622838

[192] Zhiming Liu. 2025. The ProCoS project and Duration Calculus: A personal memoir. *FACS FACTS* 2025, 2 (2025), 13–27. https://www.bcs.org/media/yd4ocehl/facs-jul25.pdf

[193] Zengyu Liu, Liqian Chen, Wanwei Liu, and Ji Wang. 2024. Synthesizing Boxes Preconditions for Deep Neural Networks. In *ISSTA 2024*. ACM, 1708–1719. doi:10.1145/3650212.3680393

[194] Zhiming Liu, Jifeng He, Xiaoshan Li, and Yifeng Chen. 2003. A Relational Model for Formal Object-Oriented Requirement Analysis in UML. In *ICFEM 2003 (Lecture Notes in Computer Science, Vol. 2885)*. Springer, 641–664. doi:10.1007/978-3-540-39893-6_36

[195] Zhiming Liu, Anders P. Ravn, Erling V. Sørensen, and Chaochen Zhou. 1993. A Probabilistic Duration Calculus. In *Responsive Computer Systems*. Springer Vienna, Vienna, 29–52. doi:10.1007/978-3-7091-9288-7_3

[196] Jian Lu. 1995. Introducing data decomposition into VDM for tractable development of programs. *SIGPLAN Not.* 30, 9 (1995), 41–50. doi:10.1145/214448.214460

[197] Jianguo Lu and Jiafu Xu. 1993. Analogical Program Derivation Based on Type Theory. *Theor. Comput. Sci.* 113, 2 (1993), 259–272. doi:10.1016/0304-3975(93)90004-D

[198] Jianan Ma, Jingyi Wang, Qi Xuan, and Zhen Wang. 2025. Provable Fairness Repair for Deep Neural Networks. In *ASE 2025*. doi:10.48550/arXiv.2104.04413 [To appear].

[199] Jianan Ma, Jingyi Wang, Qi Xuan, and Zhen Wang. 2025. Provable Repair of Deep Neural Network Defects by Preimage Synthesis and Property Refinement. In *CCS 2025*. https://arxiv.org/abs/2511.07741 [To appear].

[200] Lezhi Ma, Shangqing Liu, Yi Li, Xiaofei Xie, and Lei Bu. 2025. SpecGen: Automated Generation of Formal Program Specifications via Large Language Models. (2025), 16–28. doi:10.1109/ICSE55347.2025.00129

[201] Qian Ma, Zhenhua Duan, Nan Zhang, and Xiaobing Wang. 2015. Verification of distributed systems with the axiomatic system of MSVL. *Formal Aspects Comput.* 27, 1 (2015), 103–131. doi:10.1007/s00165-014-0303-1

[202] Wenjie Ma, Shengyuan Yang, Tian Tan, Xiaoxing Ma, Chang Xu, and Yue Li. 2023. Context Sensitivity without Contexts: A Cut-Shortcut Approach to Fast and Precise Pointer Analysis. *Proc. ACM Program. Lang.* 7, PLDI (2023), 539–564. doi:10.1145/3591242

[203] Xiwen Ma. 1982. Relational approach to formal semantics. *J. Comput.* 5, 1 (1982), 1–10.

[204] Ziyu Mao, Jingyi Wang, Jun Sun, Shengchao Qin, and Jiawen Xiong. 2025. LLM-aided Automatic Modelling for Security Protocol Verification. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 734–734. doi:10.1109/ICSE55347.2025.00197

[205] Kenneth L. McMillan. 2003. Interpolation and SAT-Based Model Checking. In *CAV 2003 (Lecture Notes in Computer Science, Vol. 2725)*. Springer, 1–13. doi:10.1007/978-3-540-45069-6_1

[206] Alan Mishchenko et al. 2012. ABC: A system for sequential synthesis and verification. http://www-cad.eecs.berkeley.edu/~alanmi/abc/. Retrieved December 1, 2025.

[207] Ben C. Moszkowski and Zohar Manna. 1983. Reasoning in Interval Temporal Logic. In *Logics of Programs (Lecture Notes in Computer Science, Vol. 164)*. Springer, 371–382. doi:10.1007/3-540-12896-4_374

[208] George C. Necula. 1997. Proof-Carrying Code. In *POPL 1997*. ACM Press, 106–119. doi:10.1145/263699.263712

[209] Mogens Nielsen, Klaus Havelund, Kim Ritter Wagner, and Chris George. 1989. The RAISE Language, Method and Tools. *Formal Aspects Comput.* 1, 1 (1989), 85–114. doi:10.1007/BF01887199

[210] Sergei Novozhilov, Mingqi Yang, Mingshuai Chen, Zhiyang Li, and Jianwei Yin. 2025. On the Almost-Sure Termination of Probabilistic Counter Programs. In *CAV 2025 (Lecture Notes in Computer Science, Vol. 15932)*. Springer, 82–104. doi:10.1007/978-3-031-98679-6_4

[211] Jonas Oberhauser, Rafael Lourenco de Lima Chehab, Diogo Behrens, Ming Fu, Antonio Paolillo, Lilith Oberhauser, Koustubha Bhat, Yuzhong Wen, Haibo Chen, Jaeho Kim, and Viktor Vafeiadis. 2021. VSync: Push-button verification and optimization for synchronization primitives on weak memory models. In *ASPLOS 2021*. ACM, 530–545. doi:10.1145/3445814.3446748

[212] Ernst-Rüdiger Olderog and Henning Dierks. 2008. *Real-time systems - formal specification and automatic verification*. Cambridge University Press. https://www.amazon.com/Real-Time-Systems-Specification-Automatic-Verification-ebook/dp/B01DM27W8A

[213] Benjamin C. Pierce. 2002. *Types and programming languages*. MIT Press. https://www.cis.upenn.edu/~bcpierce/tapl/

[214] Muhammad A. A. Pirzada, Giles Reger, Ahmed Bhayat, and Lucas C. Cordeiro. 2024. LLM-Generated Invariants for Bounded Model Checking Without Loop Unrolling. In *ASE 2024*. ACM, 1395–1407. doi:10.1145/3691620.3695512

[215] Stephen Prajna and Ali Jadbabaie. 2004. Safety verification of hybrid systems using barrier certificates. In *HSCC 2004 (Lecture Notes in Computer Science, Vol. 2993)*. Springer, 477–492. doi:10.1007/978-3-540-24743-2_32

[216] Geguang Pu, Dang Van Hung, Jifeng He, and Wang Yi. 2004. An Optimal Approach to Hardware/Software Partitioning for Synchronous Model. In *IFM 2004 (Lecture Notes in Computer Science, Vol. 2999)*. Springer, 363–381. doi:10.1007/978-3-540-24756-2_20

[217] Qi Qin, JulianAndres JiYang, Fu Song, Taolue Chen, and Xinyu Xing. 2022. DeJITLeak: eliminating JIT-induced timing side-channel leaks. In *ESEC/FSE 2022*. ACM, 872–884. doi:10.1145/3540250.3549150

[218] Shengchao Qin and Jifeng He. 2000. An algebraic approach to hardware/software partitioning. In *ICECS 2000*. IEEE, 273–276. doi:10.1109/ICECS.2000.911535

[219] Stefan Ratschan and Zhikun She. 2007. Safety verification of hybrid systems by constraint propagation-based abstraction refinement. *ACM Trans. Embed. Comput. Syst.* 6, 1 (2007), 8. doi:10.1145/1210268.1210276

[220] Anders P. Ravn. 1995. *Design of embedded real-time computing systems*. Ph. D. Dissertation. TU Lyngby. https://www.researchgate.net/publication/220689974_Design_of_embedded_real-time_computing_systems

[221] David Sanán, Yongwang Zhao, Zhe Hou, Fuyuan Zhang, Alwen Tiu, and Yang Liu. 2017. CSimpl: A Rely-Guarantee-Based Framework for Verifying Concurrent Programs. In *TACAS 2017 (Lecture Notes in Computer Science, Vol. 10205)*. 481–498. doi:10.1007/978-3-662-54577-5_28

[222] David Sanán, Yongwang Zhao, Shang-Wei Lin, and Yang Liu. 2021. CSim^2: Compositional Top-down Verification of Concurrent Systems using Rely-Guarantee. *ACM Trans. Program. Lang. Syst.* 43, 1 (2021), 2:1–2:46. doi:10.1145/3436808

[223] Mary Sheeran, Satnam Singh, and Gunnar Stålmarck. 2000. Checking Safety Properties Using Induction and a SAT-Solver. In *FMCAD 2000 (Lecture Notes in Computer Science, Vol. 1954)*. Springer, 127–144. doi:10.1007/3-540-40922-X_8

[224] Shidong Shen, Yicheng Liu, Lijun Zhang, Fu Song, and Zhilin Wu. 2024. Formal Verification of RISC-V Processor Chisel Designs. In *SETTA 2024 (Lecture Notes in Computer Science, Vol. 15469)*. Springer, 142–160. doi:10.1007/978-981-96-0602-3_8

[225] Huanhuan Sheng, Alexander Bentkamp, and Bohua Zhan. 2023. HHLPy: Practical Verification of Hybrid Systems Using Hoare Logic. In *FM 2023 (Lecture Notes in Computer Science, Vol. 14000)*. Springer, 160–178. doi:10.1007/978-3-031-27481-7_11

[226] Adnan Sherif and Jifeng He. 2002. Towards a Time Model for Circus. In *ICFEM 2002 (Lecture Notes in Computer Science, Vol. 2495)*. Springer, 613–624. doi:10.1007/3-540-36103-0_62

[227] Ziqi Shuai, Zhenbang Chen, Kelin Ma, Kunlin Liu, Yufeng Zhang, Jun Sun, and Ji Wang. 2024. Partial Solution Based Constraint Solving Cache in Symbolic Execution. *Proc. ACM Softw. Eng.* 1, FSE 2024 (2024), 2493–2514. doi:10.1145/3660817

[228] Ziqi Shuai, Zhenbang Chen, Yufeng Zhang, Jun Sun, and Ji Wang. 2021. Type and interval aware array constraint solving for symbolic execution. In *ISSTA 2021*. ACM, 361–373. doi:10.1145/3460319.3464826

[229] Andrew Sogokon, Khalil Ghorbal, Yong Kiam Tan, and André Platzer. 2018. Vector barrier certificates and comparison systems. In *FM 2018 (Lecture Notes in Computer Science, Vol. 10951)*. Springer, 418–437. doi:10.1007/978-3-319-95582-7_25

[230] Armando Solar-Lezama, Liviu Tancau, Rastislav Bodík, Sanjit A. Seshia, and Vijay A. Saraswat. 2006. Combinatorial sketching for finite programs. In *ASPLOS 2006*. ACM, 404–415. doi:10.1145/1168857.1168907

[231] J Michael Spivey and Jean-Raymond Abrial. 1992. *The Z notation*. Vol. 29. Prentice Hall Hemel Hempstead. https://dl.acm.org/doi/book/10.5555/129612

[232] Yuheng Su, Qiusong Yang, and Yiwei Ci. 2024. Predicting Lemmas in Generalization of IC3. In *DAC 2024*. ACM, 208:1–208:6. doi:10.1145/3649329.3655970

[233] Yuheng Su, Qiusong Yang, Yiwei Ci, Yingcheng Li, Tianjun Bu, and Ziyu Huang. 2025. Deeply Optimizing the SAT Solver for the IC3 Algorithm. In *CAV 2025 (Lecture Notes in Computer Science, Vol. 15931)*. Springer, 237–257. doi:10.1007/978-3-031-98668-0_12

[234] Huan Sun, David Sanán, Jingyi Wang, Yongwang Zhao, Jun Sun, and Wenhai Wang. 2025. Generalized Security-Preserving Refinement for Concurrent Systems. In *CCS 2025*. https://arxiv.org/abs/2511.06862 [To appear].

[235] Yican Sun, Hongfei Fu, Krishnendu Chatterjee, and Amir Kafshdar Goharshady. 2023. Automated Tail Bound Analysis for Probabilistic Recurrence Relations. In *CAV 2023 (Lecture Notes in Computer Science, Vol. 13966)*. Springer, 16–39. doi:10.1007/978-3-031-37709-9_2

[236] Zhihang Sun, Hongyu Fan, and Fei He. 2022. Consistency-preserving propagation for SMT solving of concurrent program verification. *Proc. ACM Program. Lang.* 6, OOPSLA2 (2022), 929–956. doi:10.1145/3563321

[237] Zewen Sun, Duanchen Xu, Yiyu Zhang, Yun Qi, Yueyang Wang, Zhiqiang Zuo, Zhaokang Wang, Yue Li, Xuandong Li, Qingda Lu, Wenwen Peng, and Shengjian Guo. 2023. BigDataflow: A Distributed Interprocedural Dataflow Analysis Framework. In *ESEC/FSE 2023*. ACM, 1431–1443. doi:10.1145/3611643.3616348

[238] Huiyu Tan, Pengfei Gao, Fu Song, Taolue Chen, and Zhilin Wu. 2024. SAT-based Formal Verification of Fault Injection Countermeasures for Cryptographic Circuits. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2024, 4 (2024), 1–39. doi:10.46586/tches.v2024.i4.1-39

[239] Qingping Tan. 1997. A higher-order unification algorithm for inductive types and dependent types. *J. Comput. Sci. Technol.* 12, 3 (1997), 231–243. doi:10.1007/BF02948973

[240] Tian Tan and Yue Li. 2023. Tai-e: A Developer-Friendly Static Analysis Framework for Java by Harnessing the Good Designs of Classics. In *ISSTA 2023*. ACM, 1093–1105. doi:10.1145/3597926.3598120

[241] Tian Tan, Yue Li, Xiaoxing Ma, Chang Xu, and Yannis Smaragdakis. 2021. Making pointer analysis more precise by unleashing the power of selective context sensitivity. *Proc. ACM Program. Lang.* 5, OOPSLA (2021), 1–27. doi:10.1145/3485524

[242] Tian Tan, Yue Li, and Jingling Xue. 2016. Making k-Object-Sensitive Pointer Analysis More Precise with Still k-Limiting. In *SAS 2016 (Lecture Notes in Computer Science, Vol. 9837)*. Springer, 489–510. doi:10.1007/978-3-662-53413-7_24

[243] Tian Tan, Yue Li, and Jingling Xue. 2017. Efficient and precise points-to analysis: modeling the heap by merging equivalent automata. In *PLDI 2017*. ACM, 278–291. doi:10.1145/3062341.3062360

[244] Wang Tan and Yi Li. 2021. Synthesis of ranking functions via DNN. *Neural Comput. Appl.* 33, 16 (2021), 9939–9959. doi:10.1007/s00521-021-05763-8

[245] Zhisong Tang. 2002. *Programming Based on Temproal Logic and Applications to Software Engineering*. Academic Press.

[246] Ren-ji Tao. 1988. Invertibility of Linear Finite Automata Over a Ring. In *ICALP 1988 (Lecture Notes in Computer Science, Vol. 317)*. Springer, 489–501. doi:10.1007/3-540-19488-6_136

[247] Thomas N. Theis and H.-S. Philip Wong. 2017. The End of Moore's Law: A New Beginning for Information Technology. *Comput. Sci. Eng.* 19, 2 (2017), 41–50. doi:10.1109/MCSE.2017.29

[248] Cong Tian and Zhenhua Duan. 2011. Expressiveness of propositional projection temporal logic with star. *Theor. Comput. Sci.* 412, 18 (2011), 1729–1744. doi:10.1016/j.tcs.2010.12.047

[249] Cong Tian and Zhenhua Duan. 2013. Detecting spurious counterexamples efficiently in abstract model checking. In *ICSE*. IEEE Computer Society, 202–211. doi:10.1109/ICSE.2013.6606566

[250] Cong Tian, Zhenhua Duan, and Zhao Duan. 2014. Making CEGAR More Efficient in Software Model Checking. *IEEE Trans. Software Eng.* 40, 12 (2014), 1206–1223. doi:10.1109/TSE.2014.2357442

[251] Wen tsun Wu. 184. Basic principles of mechanical theorem-proving in elementary geometries. *J.Sys.Sci.& Math.Scis.* (184), 207–235. doi:10.1142/9789812791085_0012 Re-published in J. Automated Reasoning, 2, 221–252, 1986, https://link.springer.com/article/10.1007/BF02328447.

[252] Wen tsun Wu. 2001. *Mathematics mechanization.* Springer Dordrecht. https://link.springer.com/book/9780792358350

[253] Alan M. Turing. 1949. *Checking a Large Routine.* Cambridge University Mathematical Laboratory, Cambridge, UK, 67–69. https://dl.acm.org/doi/10.5555/94938.94952

[254] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *NIPS 2017*. 5998–6008. https://dl.acm.org/doi/10.5555/3295222.3295349

[255] Di Wang, Jan Hoffmann, and Thomas W. Reps. 2018. PMAF: An algebraic framework for static analysis of probabilistic programs. In *PLDI 2018*. ACM, 513–528. doi:10.1145/3192366.3192408

[256] Di Wang, Jan Hoffmann, and Thomas W. Reps. 2021. Central moment analysis for cost accumulators in probabilistic programs. In *PLDI 2021*. ACM, 559–573. doi:10.1145/3453483.3454062

[257] Di Wang, Jan Hoffmann, and Thomas W. Reps. 2021. Sound probabilistic inference via guide types. In *PLDI 2021*. ACM, 788–803. doi:10.1145/3453483.3454077

[258] Di Wang, David M. Kahn, and Jan Hoffmann. 2020. Raising expectations: Automating expected cost analysis with types. *Proc. ACM Program. Lang.* 4, ICFP 2020 (2020), 110:1–110:31. doi:10.1145/3408992

[259] Di Wang and Thomas W. Reps. 2024. Newtonian Program Analysis of Probabilistic Programs. *Proc. ACM Program. Lang.* 8, OOPSLA1 (2024), 305–333. doi:10.1145/3649822

[260] Huaimin Wang and Huowang Chen. 1995. A constructor-based EI-model semantics of EI-CTRS. *J. Comput. Sci. Technol.* 10, 1 (1995), 85–96. doi:10.1007/BF02939525

[261] Ji Wang and Huowang Chen. 1992. Temporal Reasoning About Real Time Reactive Systems. In *Automated Reasoning, Proceedings of the IFIP TC12/WG12.3 International Workshop on Automated Reasoning, Beijing, P.R. China, 13–16 July 1992 (IFIP Transactions, Vol. A-19)*. North-Holland, 249–256. https://dl.acm.org/doi/10.5555/645641.664037

[262] Ji Wang and Huowang Chen. 1993. A formal technique to analyze real-time systems. In *COMPSAC 1993*. IEEE, 180–185. doi:10.1109/CMPSAC.1993.404194

[263] Ji Wang, Wei Dong, and Zhichang Qi. 2002. Slicing Hierarchical Automata for Model Checking UML Statecharts. In *ICFEM 2002 (Lecture Notes in Computer Science, Vol. 2495)*. Springer, 435–446. doi:10.1007/3-540-36103-0_45

[264] Jiawan Wang, Wenxia Liu, Muzimiao Zhang, Jiaqi Wei, Yuhui Shi, Lei Bu, and Xuandong Li. 2024. Scenario-Based Flexible Modeling and Scalable Falsification for Reconfigurable CPSs. In *CAV 2024 (Lecture Notes in Computer Science, Vol. 14683)*. Springer, 329–355. doi:10.1007/978-3-031-65633-0_15

[265] Jinyi Wang, Yican Sun, Hongfei Fu, Krishnendu Chatterjee, and Amir Kafshdar Goharshady. 2021. Quantitative analysis of assertion violations in probabilistic programs. In *PLDI 2021*. ACM, 1171–1186. doi:10.1145/3453483.3454102

[266] Jiayi Wang, Yu Wang, Ke Wang, and Linzhang Wang. 2025. SILVA: A Scalable Incremental Layered Sparse Value-Flow Analysis. *ACM Trans. Softw. Eng. Methodol.* (2025). doi:10.1145/3725214 https://doi.org/10.1145/3725214.

[267] Ji Wang, Naijun Zhan, Xinyu Feng, and Zhiming Liu. 2019. Overview of Formal Methods. *Ruan Jian Xue Bao / Journal of Software* 30, 1 (2019), 33–61. http://www.jos.org.cn/1000-9825/5652.htm (In Chinese).

[268] Kai Wang, Aftab Hussain, Zhiqiang Zuo, Guoqing Xu, and Ardalan Amiri Sani. 2017. Graspan: A Single-machine Disk-based Graph System for Interprocedural Static Analyses of Large-scale Systems Code. In *ASPLOS 2017*. ACM, 389–404. doi:10.1145/3093336.3037744

[269] Kun Wang, Jingyi Wang, Christopher M Poskitt, Xiangxiang Chen, Jun Sun, and Peng Cheng. 2023. K-ST: A formal executable semantics of the structured text language for PLCs. *IEEE Transactions on Software Engineering* 49, 10 (2023), 4796–4813. doi:10.1109/TSE.2023.3315292

[270] Lingtai Wang, Naijun Zhan, and Jie An. 2018. The Opacity of Real-Time Automata. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 37, 11 (2018), 2845–2856. doi:10.1109/TCAD.2018.2857363

[271] Peixin Wang, Hongfei Fu, Krishnendu Chatterjee, Yuxin Deng, and Ming Xu. 2020. Proving expected sensitivity of probabilistic programs with randomized variable-dependent termination time. *Proc. ACM Program. Lang.* 4, POPL (2020), 25:1–25:30. doi:10.1145/3371093

[272] Peixin Wang, Hongfei Fu, Amir Kafshdar Goharshady, Krishnendu Chatterjee, Xudong Qin, and Wenjun Shi. 2019. Cost analysis of nondeterministic probabilistic programs. In *PLDI 2019*. ACM, 204–220. doi:10.1145/3314221.3314581

[273] Peixin Wang, Tengshun Yang, Hongfei Fu, Guanyan Li, and C.-H. Luke Ong. 2024. Static Posterior Inference of Bayesian Probabilistic Programming via Polynomial Solving. *Proc. ACM Program. Lang.* 8, PLDI (2024), 1361–1386. doi:10.1145/3656432

[274] Qiuye Wang, Mingshuai Chen, Bai Xue, Naijun Zhan, and Joost-Pieter Katoen. 2022. Encoding inductive invariants as barrier certificates: Synthesis via difference-of-convex programming. *Inf. Comput.* 289, Part (2022), 104965. doi:10.1016/j.ic.2022.104965

[275] Qiuye Wang, Mingshuai Chen, Bai Xue, Naijun Zhan, and Joost-Pieter Katoen. 2021. Synthesizing invariant barrier certificates via difference-of-convex programming. In *CAV 2021 (1) (Lecture Notes in Computer Science, Vol. 12759)*. Springer, 443–466. doi:10.1007/978-3-030-81685-8_21

[276] Shuling Wang, Zekun Ji, Xiong Xu, Bohua Zhan, Qiang Gao, and Naijun Zhan. 2024. Formally Verified C Code Generation from Hybrid Communicating Sequential Processes. In *ICCPS 2024*. IEEE, 123–134. doi:10.1109/ICCPS61052.2024.00018

[277] Shuling Wang, Naijun Zhan, and Liang Zou. 2015. An Improved HHL Prover: An Interactive Theorem Prover for Hybrid Systems. In *ICFEM 2015 (Lecture Notes in Computer Science, Vol. 9407)*. Springer, 382–399. doi:10.1007/978-3-319-25423-4_25

[278] Xianchang Wang and Huowang Chen. 1991. On Semantics of TMS. In *IJCAI 1991*. Morgan Kaufmann. https://dl.acm.org/doi/10.5555/1631171.1631217

[279] Xizao Wang, Zhiqiang Zuo, Lei Bu, and Jianhua Zhao. 2023. DStream: A Streaming-Based Highly Parallel IFDS Framework. In *ICSE 2023*. IEEE, 2488–2500. doi:10.1109/ICSE48619.2023.00208

[280] Yu Wang, Ke Wang, Fengjuan Gao, and Linzhang Wang. 2020. Learning semantic program embeddings with graph interval neural network. *Proc. ACM Program. Lang.* 4, OOPSLA (2020), 137:1–137:27. doi:10.1145/3428205

[281] Yuting Wang, Xiangzhe Xu, Pierre Wilke, and Zhong Shao. 2020. CompCertELF: verified separate compilation of C programs into ELF object files. *Proc. ACM Program. Lang.* 4, OOPSLA (2020), 197:1–197:28. doi:10.1145/3428265

[282] Yuting Wang, Ling Zhang, Zhong Shao, and Jérémie Koenig. 2022. Verified compilation of C programs with a nominal memory model. *Proc. ACM Program. Lang.* 6, POPL (2022), 1–31. doi:10.1145/3498686

[283] Zhongyi Wang, Tengjie Lin, Mingshuai Chen, Mingqi Yang, Haokun Li, Xiao Yi, Shengchao Qin, and Jianwei Yin. 2025. Preguss: It Analyzes, It Specifies, It Verifies. In *LMPL 2025*. ACM, 118–123. doi:10.1145/3759425.3763394

[284] Zhaoguo Wang, Zhou Zhou, Yicun Yang, Haoran Ding, Gansen Hu, Ding Ding, Chuzhe Tang, Haibo Chen, and Jinyang Li. 2022. WeTune: Automatic Discovery and Verification of Query Rewrite Rules. In *SIGMOD 2022*. ACM, 94–107. doi:10.1145/3514221.3526125

[285] Cheng Wen, Jialun Cao, Jie Su, Zhiwu Xu, Shengchao Qin, Mengda He, Haokun Li, Shing-Chi Cheung, and Cong Tian. 2024. Enchanting Program Specification Synthesis by Large Language Models Using Static Analysis and Program Verification. In *CAV 2024 (2) (Lecture Notes in Computer Science, Vol. 14682)*. Springer, 302–328. doi:10.1007/978-3-031-65630-9_16

[286] Guangyuan Wu, Weining Cao, Yuan Yao, Hengfeng Wei, Taolue Chen, and Xiaoxing Ma. 2024. LLM Meets Bounded Model Checking: Neuro-symbolic Loop Invariant Inference. In *ASE 2024*, Vladimir Filkov, Baishakhi Ray, and Minghui Zhou (Eds.). ACM, 406–417. doi:10.1145/3691620.3695014

[287] Guangyuan Wu, Weining Cao, Yuan Yao, Hengfeng Wei, Taolue Chen, and Xiaoxing Ma. 2024. LLM Meets Bounded Model Checking: Neuro-Symbolic Loop Invariant Inference. In *ASE 2024*. ACM, 406–417. doi:10.1145/3691620.3695014

[288] Haoze Wu, Clark W. Barrett, and Nina Narodytska. 2024. Lemur: Integrating Large Language Models in Automated Program Verification. In *ICLR 2024*. https://openreview.net/forum?id=Q3YaCghZNt

[289] Hao Wu, Yu-Fang Chen, Zhilin Wu, Bican Xia, and Naijun Zhan. 2024. A decision procedure for string constraints with string/integer conversion and flat regular constraints. *Acta Informatica* 61, 1 (2024), 23–52. doi:10.1007/s00236-023-00446-4

[290] Huiling Wu, Anran Cui, and Yuxin Deng. 2024. An Assertion-Based Logic for Local Reasoning about Probabilistic Programs. In *SETTA 2024 (Lecture Notes in Computer Science, Vol. 15469)*. Springer, 25–45. doi:10.1007/978-981-96-0602-3_2

[291] Hao Wu, Jie Wang, Bican Xia, Xiakun Li, Naijun Zhan, and Ting Gan. 2024. Nonlinear Craig Interpolant Generation Over Unbounded Domains by Separating Semialgebraic Sets. In *FM 2024 (Lecture Notes in Computer Science, Vol. 14933)*. Springer, 92–110. doi:10.1007/978-3-031-71162-6_5

[292] Hao Wu, Qiuye Wang, Bai Xue, Naijun Zhan, Lihong Zhi, and Zhi-Hong Yang. 2025. Synthesizing Invariants for Polynomial Programs by Semidefinite Programming. *ACM Trans. Program. Lang. Syst.* 47, 1 (2025), 1:1–1:35. doi:10.1145/3708559

[293] Xueguang Wu, Liqian Chen, Antoine Miné, Wei Dong, and Ji Wang. 2016. Static Analysis of Runtime Errors in Interrupt-Driven Programs via Sequentialization. *ACM Trans. Embed. Comput. Syst.* 15, 4 (2016), 70:1–70:26. doi:10.1145/2914789

[294] Xiwei Wu, Yueyang Feng, Xiaoyang Lu, Tianchuan Lin, Kan Liu, Zhiyi Wang, Shushu Wu, Lihan Xie, Chengxi Yang, Hongyi Zhong, Naijun Zhan, Zhenjiang Hu, and Qinxiang Cao. 2025. QCP: A Practical Separation Logic-based C Program Verification Tool. *CoRR* abs/2505.12878 (2025). doi:10.48550/arXiv.2505.12878

[295] Yulun Wu, Bohua Zhan, and Bican Xia. 2024. OSVAuto: Semi-automatic verifier for functional specifications of operating systems. *CoRR* abs/2403.13457 (2024). doi:10.48550/arXiv.2403.13457

[296] Bican Xia. 2007. DISCOVERER: A tool for solving semi-algebraic systems. *ACM Commun. Comput. Algebra* 41, 3 (2007), 102–103. doi:10.1145/1358190.1358197

[297] Bican Xia and Lu Yang. 2002. An Algorithm for Isolating the Real Solutions of Semi-algebraic Systems. *J. Symb. Comput.* 34, 5 (2002), 461–477. doi:10.1006/jsco.2002.0572

[298] Yechuan Xia, Anna Becchi, Alessandro Cimatti, Alberto Griggio, Jianwen Li, and Geguang Pu. 2023. Searching for *i*-Good Lemmas to Accelerate Safety Model Checking. In *CAV 2023 (Lecture Notes in Computer Science, Vol. 13965)*. Springer, 288–308. doi:10.1007/978-3-031-37703-7_14

[299] Dingbao Xie, Lei Bu, and Xuandong Li. 2014. Deriving Unbounded Proof of Linear Hybrid Automata from Bounded Verification. In *RTSS 2014*. IEEE Computer Society, 128–137. doi:10.1109/RTSS.2014.22

[300] Dingbao Xie, Lei Bu, Jianhua Zhao, and Xuandong Li. 2014. SAT-LP-IIS joint-directed path-oriented bounded reachability analysis of linear hybrid automata. *Formal Methods Syst. Des.* 45, 1 (2014), 42–62. doi:10.1007/s10703-014-0210-3

[301] Dingbao Xie, Wen Xiong, Lei Bu, and Xuandong Li. 2017. Deriving Unbounded Reachability Proof of Linear Hybrid Automata during Bounded Checking Procedure. *IEEE Trans. Computers* 66, 3 (2017), 416–430. doi:10.1109/TC.2016.2604308

[302] Ma Xiwen and Guo Weide. 1983. W-JS: A Modal Logic of Knowledge. In *IJCAI 1983*. 398–401. https://www.ijcai.org/Proceedings/83-1/Papers/094.pdf

[303] Baowen Xu, Yingzhou Zhang, and Yanhui Li. 2004. Retrospect and Prospect of Formal Methods Education in China. In *TFM 2004 (Lecture Notes in Computer Science, Vol. 3294)*. Springer, 225–234. doi:10.1007/978-3-540-30472-2_15

[304] Fengwei Xu, Ming Fu, Xinyu Feng, Xiaoran Zhang, Hui Zhang, and Zhaohui Li. 2016. A Practical Verification Framework for Preemptive OS Kernels. In *CAV 2016 (Lecture Notes in Computer Science, Vol. 9780)*. Springer, 59–79. doi:10.1007/978-3-319-41540-6_4

[305] Jiafu Xu (Ed.). 1983. *Programming Languages for System Software.* China Science Publishing & Media Ltd. in Chinese.

[306] Jiafu Xu, Daoxu Chen, Jian Lu, and Zhijian Wang (Eds.). 1994. *Software Automation.* Tsinghua University Press. in Chinese.

[307] Jiafu Xu and Jian Lu (Eds.). 2000. *Software Programming Languages and Implementations.* China Science Publishing & Media Ltd. in Chinese.

[308] Jiafu Xu and Fangming Song (Eds.). 2013. *Quantum Programming Languages.* China Science Publishing & Media Ltd. in Chinese.

[309] Ming Xu and Yuxin Deng. 2020. Time-bounded termination analysis for probabilistic programs with delays. *Inf. Comput.* 275 (2020), 104634. doi:10.1016/j.ic.2020.104634

[310] Rongchen Xu, Jianhui Chen, and Fei He. 2022. Data-Driven Loop Bound Learning for Termination Analysis. In *ICSE 2022*. ACM, 499–510. doi:10.1145/3510003.3510220

[311] Rongchen Xu, Fei He, and Bow-Yaw Wang. 2020. Interval counterexamples for loop invariant learning. In *ESEC/SIGSOFT FSE 2020*. ACM, 111–122. doi:10.1145/3368089.3409752

[312] Wenjing Xu, Yongwang Zhao, Chengtao Cao, Jean Raphael Ngnie Sighom, Lei Wang, Zhe Jiang, and Shihong Zou. 2021. Apply Formal Methods in Certifying the SyberX High-Assurance Kernel. In *FM 2021 (Lecture Notes in Computer Science, Vol. 13047)*. Springer, 788–798. doi:10.1007/978-3-030-90870-6_46

[313] Xiong Xu, Ehsan Ahmad, Shuling Wang, Xiangyu Jin, Bohua Zhan, and Naijun Zhan. 2025. Modeling and Verification of Hybrid Systems by Extending AADL. *ACM Trans. Softw. Eng. Methodol.* (2025). doi:10.1145/3737698

[314] Xiong Xu, Jean-Pierre Talpin, Shuling Wang, Hao Wu, Bohua Zhan, Xinxin Liu, and Naijun Zhan. 2025. HpC: A Calculus for Hybrid and Mobile Systems. *Proc. ACM Program. Lang.* 9, OOPSLA1 (2025), 1158–1183. doi:10.1145/3720478

[315] Xiong Xu, Jean-Pierre Talpin, Shuling Wang, Bohua Zhan, and Naijun Zhan. 2023. Semantics Foundation for Cyber-physical Systems Using Higher-order UTP. *ACM Trans. Softw. Eng. Methodol.* 32, 1 (2023), 9:1–9:48. doi:10.1145/3517192

[316] Xiong Xu, Shuling Wang, Bohua Zhan, Xiangyu Jin, Jean-Pierre Talpin, and Naijun Zhan. 2022. Unified graphical co-modeling, analysis and verification of cyber-physical systems by combining AADL and Simulink/Stateflow. *Theor. Comput. Sci.* 903 (2022), 1–25. doi:10.1016/j.tcs.2021.11.008

[317] Xiong Xu, Bohua Zhan, Shuling Wang, Jean-Pierre Talpin, and Naijun Zhan. 2023. A denotational semantics of Simulink with higher-order UTP. *J. Log. Algebraic Methods Program.* 130 (2023), 100809. doi:10.1016/j.jlamp.2022.100809

[318] Bai Xue, Martin Fränzle, and Naijun Zhan. 2018. Under-Approximating Reach Sets for Polynomial Continuous Systems. In *HSCC 2018*. ACM, 51–60. doi:10.1145/3178126.3178133

[319] Bai Xue, Martin Fränzle, and Naijun Zhan. 2020. Inner-Approximating Reachable Sets for Polynomial Systems With Time-Varying Uncertainties. *IEEE Trans. Autom. Control.* 65, 4 (2020), 1468–1483. doi:10.1109/TAC.2019.2923049

[320] Bai Xue, Martin Fränzle, Naijun Zhan, Sergiy Bogomolov, and Bican Xia. 2020. Safety Verification for Random Ordinary Differential Equations. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 39, 11 (2020), 4090–4101. doi:10.1109/TCAD.2020.3013135

[321] Bai Xue, Peter Nazier Mosaad, Martin Fränzle, Mingshuai Chen, Yangjia Li, and Naijun Zhan. 2017. Safe Over- and Under-Approximation of Reachable Sets for Delay Differential Equations. In *FORMATS 2017 (Lecture Notes in Computer Science, Vol. 10419)*. Springer, 281–299. doi:10.1007/978-3-319-65765-3_16

[322] Bai Xue, Zhikun She, and Arvind Easwaran. 2016. Under-Approximating Backward Reachable Sets by Polytopes. In *CAV 2016 (Lecture Notes in Computer Science, Vol. 9779)*. Springer, 457–476. doi:10.1007/978-3-319-41528-4_25

[323] Bai Xue, Qiuye Wang, Shenghua Feng, and Naijun Zhan. 2021. Over- and Underapproximating Reach Sets for Perturbed Delay Differential Equations. *IEEE Trans. Autom. Control.* 66, 1 (2021), 283–290. doi:10.1109/TAC.2020.2977993

[324] Bai Xue, Miaomiao Zhang, Arvind Easwaran, and Qin Li. 2020. PAC Model Checking of Black-Box Continuous-Time Dynamical Systems. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 39, 11 (2020), 3944–3955. doi:10.1109/TCAD.2020.3012251

[325] Gaogao Yan, Li Jiao, Shuling Wang, Lingtai Wang, and Naijun Zhan. 2020. Automatically Generating SystemC Code from HCSP Formal Models. *ACM Trans. Softw. Eng. Methodol.* 29, 1 (2020), 4:1–4:39. doi:10.1145/3360002

[326] Zhiyuan Yan, Wenji Fang, Mengming Li, Min Li, Shang Liu, Zhiyao Xie, and Hongce Zhang. 2025. AssertLLM: Generating Hardware Verification Assertions from Design Specifications via Multi-LLMs. In *ASPDAC 2025*. ACM, 614–621. doi:10.1145/3658617.3697756
[327] Kai Yang, Zhenhua Duan, Cong Tian, and Nan Zhang. 2018. A compiler for MSVL and its applications. *Theor. Comput. Sci.* 749 (2018), 2–16. doi:10.1016/j.tcs.2017.07.032
[328] Lu Yang. 1999. Recent Advances on Determining the Number of Real Roots of Parametric Polynomials. *J. Symb. Comput.* 28, 1–2 (1999), 225–242. doi:10.1006/jsco.1998.0274
[329] Pengfei Yang, Jianlin Li, Jiangchao Liu, Cheng-Chao Huang, Renjue Li, Liqian Chen, Xiaowei Huang, and Lijun Zhang. 2021. Enhancing Robustness Verification for Deep Neural Networks via Symbolic Propagation. *Formal Aspects Comput.* 33, 3 (2021), 407–435. doi:10.1007/s00165-021-00548-1
[330] Yang Yang, Lei Bu, and Xuandong Li. 2012. Forward and backward: Bounded model checking of linear hybrid automata from two directions. In *FMCAD 2012*. IEEE, 204–208. https://ieeexplore.ieee.org/document/6462575
[331] Zhengfeng Yang, Wang Lin, and Min Wu. 2015. Exact Safety Verification of Hybrid Systems Based on Bilinear SOS Representation. *ACM Trans. Embed. Comput. Syst.* 14, 1 (2015), 16:1–16:19. doi:10.1145/2629424
[332] Bo Yi and Jiafu Xu. 1993. Analogy Calculus. *Theor. Comput. Sci.* 113, 2 (1993), 211–230. doi:10.1016/0304-3975(93)90002-B
[333] Xiaodong Yi, Ji Wang, and Xuejun Yang. 2005. Verification of C Programs using Slicing Execution. In *QSIC 2005*. IEEE Computer Society, 109–116. doi:10.1109/QSIC.2005.72
[334] Xiaodong Yi, Ji Wang, and Xuejun Yang. 2006. Stateful Dynamic Partial-Order Reduction. In *ICFEM 2006 (Lecture Notes in Computer Science, Vol. 4260)*. Springer, 149–167. doi:10.1007/11901433_9
[335] Liangze Yin, Wei Dong, Wanwei Liu, Yunchou Li, and Ji Wang. 2018. YOGAR-CBMC: CBMC with Scheduling Constraint Based Abstraction Refinement - (Competition Contribution). In *TACAS 2018 (Lecture Notes in Computer Science, Vol. 10806)*. Springer, 422–426. doi:10.1007/978-3-319-89963-3_25
[336] Liangze Yin, Wei Dong, Wanwei Liu, and Ji Wang. 2018. Scheduling constraint based abstraction refinement for weak memory models. In *ASE 2018*. ACM, 645–655. doi:10.1145/3238147.3238223
[337] Liangze Yin, Wei Dong, Wanwei Liu, and Ji Wang. 2019. Parallel refinement for multi-threaded program verification. In *ICSE 2019*. IEEE / ACM, 643–653. doi:10.1109/ICSE.2019.00074
[338] Liangze Yin, Wei Dong, Wanwei Liu, and Ji Wang. 2020. On Scheduling Constraint Abstraction for Multi-Threaded Program Verification. *IEEE Trans. Software Eng.* 46, 5 (2020), 549–565. doi:10.1109/TSE.2018.2864122
[339] Liangze Yin, Yiwei Li, Kun Chen, Wei Dong, and Ji Wang. 2025. Incremental Verification of Concurrent Programs through Refinement Constraint Adaptation. *Proc. ACM Softw. Eng.* 2, ISSTA (2025), 2251–2272. doi:10.1145/3728976
[340] Mingsheng Ying. 2001. *Topology in process calculus - approximate correctness and infinite evolution of concurrent programs.* Springer. doi:10.1007/978-1-4613-0123-3
[341] Mingsheng Ying. 2002. Additive models of probabilistic processes. *Theor. Comput. Sci.* 275, 1–2 (2002), 481–519. doi:10.1016/S0304-3975(01)00294-8
[342] Mingsheng Ying. 2011. Floyd-Hoare logic for quantum programs. *ACM Trans. Program. Lang. Syst.* 33, 6 (2011), 19:1–19:49. doi:10.1145/2049706.2049708
[343] Mingsheng Ying. 2024. *Foundations of quantum programming, 2nd Edition.* Elsevier. https://www.sciencedirect.com/book/monograph/9780443159428/foundations-of-quantum-programming
[344] Mingsheng Ying, Shenggang Ying, and Xiaodi Wu. 2017. Invariants of quantum programs: characterisations and generation. (2017), 818–832. doi:10.1145/3009837.3009840
[345] Mingsheng Ying, Li Zhou, Yangjia Li, and Yuan Feng. 2022. A proof system for disjoint parallel quantum programs. *Theor. Comput. Sci.* 897 (2022), 164–184. doi:10.1016/j.tcs.2021.10.025
[346] Hengbiao Yu, Zhenbang Chen, Xianjin Fu, Ji Wang, Zhendong Su, Jun Sun, Chun Huang, and Wei Dong. 2020. Symbolic verification of message passing interface programs. In *ICSE 2020*. ACM, 1248–1260. doi:10.1145/3377811.3380419
[347] Hengbiao Yu, Zhenbang Chen, Ji Wang, Zhendong Su, and Wei Dong. 2018. Symbolic verification of regular properties. In *ICSE 2018*. ACM, 871–881. doi:10.1145/3180155.3180227
[348] Qianshan Yu, Fei He, and Bow-Yaw Wang. 2020. Incremental predicate analysis for regression verification. *Proc. ACM Program. Lang.* 4, OOPSLA (2020), 184:1–184:25. doi:10.1145/3428252
[349] Shiwen Yu, Zengyu Liu, Ting Wang, and Ji Wang. 2024. Neural Solving Uninterpreted Predicates with Abstract Gradient Descent. *ACM Trans. Softw. Eng. Methodol.* 33, 8 (2024), 215:1–215:47. doi:10.1145/3675394
[350] Shiwen Yu, Ting Wang, and Ji Wang. 2023. Loop Invariant Inference through SMT Solving Enhanced Reinforcement Learning. In *ISSTA 2023*. ACM, 175–187. doi:10.1145/3597926.3598047
[351] Xinyao Yu, Ji Wang, Chaochen Zhou, and Paritosh K. Pandya. 1994. Formal Design of Hybrid Systems. In *FTRTFT 1994 (Lecture Notes in Computer Science, Vol. 863)*. Springer, 738–755. doi:10.1007/3-540-58468-4_193
[352] Chongyi Yuan. 2025. *Principle of Petri Nets.* Springer. doi:10.1007/978-981-97-7336-7

[353] Richard Zach. 2023. Hilbert's Program. In *The Stanford Encyclopedia of Philosophy* (Winter 2023 ed.). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/win2023/entries/hilbert-program/

[354] Xia Zeng, Wang Lin, Zhengfeng Yang, Xin Chen, and Lilei Wang. 2016. Darboux-type barrier certificates for safety verification of nonlinear hybrid systems. In *EMSOFT 2016*. ACM, 1–10. doi:10.1145/2968478.2968484

[355] Junpeng Zha, Hongjin Liang, and Xinyu Feng. 2022. Verifying optimizations of concurrent programs in the promising semantics. In *PLDI 2022*. ACM, 903–917. doi:10.1145/3519939.3523734

[356] Bohua Zhan. 2016. AUTO2, A Saturation-Based Heuristic Prover for Higher-Order Logic. In *ITP 2016 (Lecture Notes in Computer Science, Vol. 9807)*. Springer, 441–456. doi:10.1007/978-3-319-43144-4_27

[357] Bohua Zhan. 2022. User Interface Design in the HolPy Theorem Prover (Invited Talk). In *ITP 2022 (LIPIcs, Vol. 237)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2:1–2:1. doi:10.4230/LIPIcs.ITP.2022.2

[358] Naijun Zhan. 2000. A higher-order duration calculus and its completeness. *Science in China* 46, 6 (2000), 625–640. https://www.sciengine.com/cfs/files/pdfs/view/1006-9321/RJeu53fZxmiRvSWXA.pdf

[359] Naijun Zhan, Shuling Wang, and Hengjun Zhao (Eds.). 2017. *Formal Verification of Simulink/Stateflow Diagrams, A Deductive Approach*. Springer. doi:10.1007/978-3-319-47016-0

[360] Naijun Zhan, Bohua Zhan, Shuling Wang, Dimitar P. Guelev, and Xiangyu Jin. 2023. A Generalized Hybrid Hoare Logic. *CoRR* abs/2303.15020 (2023). doi:10.48550/arXiv.2303.15020

[361] Chenxi Zhang, Yufei Liang, Tian Tan, Chang Xu, Shuangxiang Kan, Yulei Sui, and Yue Li. 2025. Interactive Cross-Language Pointer Analysis for Resolving Native Code in Java Programs. In *ICSE 2025*. IEEE, 1089–1100. doi:10.1109/ICSE55347.2025.00075

[362] Feng Zhang, Leping Zhang, Yongwang Zhao, Yang Liu, and Jun Sun. 2023. Refinement-based Specification and Analysis of Multi-core ARINC 653 Using Event-B. *Formal Aspects Comput.* 35, 4 (2023), 24:1–24:29. doi:10.1145/3617183

[363] Jun Zhang, Pengfei Gao, Fu Song, and Chao Wang. 2018. SCInfer: Refinement-Based Verification of Software Countermeasures Against Side-Channel Attacks. In *CAV 2018*, Vol. 10982. Springer, 157–177. doi:10.1007/978-3-319-96142-2_12

[364] Jingzhong Zhang, Lu Yang, and Mike Deng. 1990. The Parallel Numerical Method of Mechanical Theorem Proving. *Theor. Comput. Sci.* 74, 3 (1990), 253–271. doi:10.1016/0304-3975(90)90077-U

[365] Jing-Zhong Zhang, Shang-Ching Chou, and Xiao-Shan Gao. 1995. Automated production of traditional proofs for theorems in Euclidean geometry I. The Hilbert intersection point theorems. *Ann Math Artif Intell* 13 (1995), 109–137. doi:10.1007/BF01531326

[366] Ling Zhang, Yuting Wang, Yalun Liang, and Zhong Shao. 2025. CompCertOC: Verified Compositional Compilation of Multi-threaded Programs with Shared Stacks. *Proc. ACM Program. Lang.* 9, PLDI (2025), 651–674. doi:10.1145/3729276

[367] Shuo Zhang, Fei He, and Ming Gu. 2015. VeRV: A temporal and data-concerned verification framework for the vehicle bus systems. In *INFOCOM 2015*. IEEE, 1167–1175. doi:10.1109/INFOCOM.2015.7218491

[368] Teng Zhang, Yufei Liang, Ganlin Li, Tian Tan, Chang Xu, and Yue Li. 2025. Bridge the Islands: Pointer Analysis for Microservice Systems. *Proc. ACM Softw. Eng.* 2, ISSTA (2025), 504–526. doi:10.1145/3728896

[369] Yedi Zhang, Yufan Cai, Xinyue Zuo, Xiaokun Luan, Kailong Wang, Zhe Hou, Yifan Zhang, Zhiyuan Wei, Meng Sun, Jun Sun, et al. 2025. Position: Trustworthy AI Agents Require the Integration of Large Language Models and Formal Methods. In *ICML 2025*. https://icml.cc/virtual/2025/poster/40101 [To appear].

[370] Yufeng Zhang, Zhenbang Chen, Ziqi Shuai, Tianqi Zhang, Kenli Li, and Ji Wang. 2020. Multiplex Symbolic Execution: Exploring Multiple Paths by Solving Once. In *ASE 2020*. IEEE, 846–857. doi:10.1145/3324884.3416645

[371] Yufeng Zhang, Zhenbang Chen, Ji Wang, Wei Dong, and Zhiming Liu. 2015. Regular Property Guided Dynamic Symbolic Execution. In *ICSE 2015*. IEEE Computer Society, 643–653. doi:10.1109/ICSE.2015.80

[372] Yu Zhang, Jérémie Koenig, Zhong Shao, and Yuting Wang. 2025. Unifying Compositional Verification and Certified Compilation with a Three-Dimensional Refinement Algebra. *Proc. ACM Program. Lang.* 9, POPL (2025), 1903–1933. doi:10.1145/3704900

[373] Yuhao Zhang, Luyao Ren, Liqian Chen, Yingfei Xiong, Shing-Chi Cheung, and Tao Xie. 2020. Detecting numerical bugs in neural network architectures. In *ESEC/FSE 2020*. ACM, 826–837. doi:10.1145/3368089.3409720

[374] Zhaodi Zhang, Yiting Wu, Si Liu, Jing Liu, and Min Zhang. 2022. Provably tightest linear approximation for robustness verification of sigmoid-like neural networks. In *ASE 2022*. 1–13. doi:10.1145/3551349.3556907

[375] Hanrui Zhao, Banglong Liu, Lydia Dehbi, Huijiao Xie, Zhengfeng Yang, and Haifeng Qian. 2024. Polynomial Neural Barrier Certificate Synthesis of Hybrid Systems via Counterexample Guidance. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 43, 11 (2024), 3756–3767. doi:10.1109/TCAD.2024.3447226

[376] Hanrui Zhao, Niuniu Qi, Lydia Dehbi, Xia Zeng, and Zhengfeng Yang. 2023. Formal Synthesis of Neural Barrier Certificates for Continuous Systems via Counterexample Guided Learning. *ACM Trans. Embed. Comput. Syst.* 22, 5s (2023), 146:1–146:21. doi:10.1145/3609125

[377] Hanrui Zhao, Niuniu Qi, Lydia Dehbi, Xia Zeng, and Zhengfeng Yang. 2023. Formal Synthesis of Neural Barrier Certificates for Continuous Systems via Counterexample Guided Learning. *ACM Trans. Embed. Comput. Syst.* 22, 5s (2023), 146:1–146:21. doi:10.1145/3609125

[378] Hengjun Zhao, Mengfei Yang, Naijun Zhan, Bin Gu, Liang Zou, and Yao Chen. 2014. Formal Verification of a Descent Guidance Control Program of a Lunar Lander. In *FM 2014 (Lecture Notes in Computer Science, Vol. 8442)*. Springer, 733–748. doi:10.1007/978-3-319-06410-9_49

[379] Hengjun Zhao, Xia Zeng, Taolue Chen, and Zhiming Liu. 2020. Synthesizing barrier certificates using neural networks. In *HSCC 2020*. ACM, 25:1–25:11. doi:10.1145/3365365.3382222

[380] Hengjun Zhao, Xia Zeng, Taolue Chen, Zhiming Liu, and Jim Woodcock. 2021. Learning safe neural network controllers with barrier certificates. *Formal Aspects Comput.* 33, 3 (2021), 437–455. doi:10.1007/978-3-030-62822-2_11

[381] Jianing Zhao, Shaoyuan Li, and Xiang Yin. 2024. A unified framework for verification of observational properties for partially-observed discrete-event systems. *IEEE Trans. Automat. Control* 69, 7 (2024), 4710–4717. doi:10.1109/TAC.2024.3355378

[382] Yongwang Zhao and David Sanán. 2019. Rely-Guarantee Reasoning About Concurrent Memory Management in Zephyr RTOS. In *CAV 2019 (Lecture Notes in Computer Science, Vol. 11562)*. Springer, 515–533. doi:10.1007/978-3-030-25543-5_29

[383] Yongwang Zhao, Zhibin Yang, David Sanán, and Yang Liu. 2015. Event-based formalization of safety-critical operating system standards: An experience report on ARINC 653 using Event-B. In *ISSRE 2015*. IEEE Computer Society, 281–292. doi:10.1109/ISSRE.2015.7381821

[384] Dapeng Zhi, Peixin Wang, Si Liu, C-H Luke Ong, and Min Zhang. 2024. Unifying qualitative and quantitative safety verification of DNN-controlled systems. In *CAV 2024*. Springer, 401–426. doi:10.1007/978-3-031-65630-9_20

[385] Chaochen Zhou. 1982. Weakest environment of communicating processes. In *American Federation of Information Processing Societies: 1982 National Computer Conference, 7–10 June, 1982, Houston, Texas, USA (AFIPS Conference Proceedings, Vol. 51)*. AFIPS Press, 679–690. doi:10.1145/1500774.1500860

[386] Chaochen Zhou. 1994. Linear Duration Invariants. In *FTRTFT 1994 (Lecture Notes in Computer Science, Vol. 863)*. Springer, 86–109. doi:10.1007/3-540-58468-4_161

[387] Chaochen Zhou, Dimitar P. Guelev, and Naijun Zhan. 1999. A higher-order duration calculus. In *The Symposium in Celebration of the Work of C. A. R. Hoare*. Prentice Hall International (UK) Ltd. https://www.researchgate.net/publication/2298859_A_Higher-Order_Duration_Calculus

[388] Chaochen Zhou and Michael R. Hansen. 2004. *Duration Calculus - A Formal Approach to Real-Time Systems*. Springer. doi:10.1007/978-3-662-06784-0

[389] Chaochen Zhou, Michael R. Hansen, and Peter Sestoft. 1993. Decidability and Undecidability Results for Duration Calculus. In *STACS 1993 (Lecture Notes in Computer Science, Vol. 665)*. Springer, 58–68. doi:10.1007/3-540-56503-5_8

[390] Chaochen Zhou, C. A. R. Hoare, and Anders P. Ravn. 1991. A Calculus of Durations. *Inf. Process. Lett.* 40, 5 (1991), 269–276. doi:10.1016/0020-0190(91)90122-X

[391] Chaochen Zhou, Dang Van Hung, and Xiaoshan Li. 1995. A Duration Calculus with Infinite Intervals. In *FCT 1995 (Lecture Notes in Computer Science, Vol. 965)*. Springer, 16–41. doi:10.1007/3-540-60249-6_39

[392] Chaochen Zhou and Xiaoshan Li. 1994. *A mean value calculus of durations*. Prentice Hall International (UK) Ltd., GBR, 431–451. https://dl.acm.org/doi/10.5555/197600.197633

[393] Chaochen Zhou, Anders P. Ravn, and Michael R. Hansen. 1992. An Extended Duration Calculus for Hybrid Real-Time Systems. In *Hybrid Systems (Lecture Notes in Computer Science, Vol. 736)*. Springer, 36–59. doi:10.1007/3-540-57318-6_23

[394] Chaochen Zhou, Ji Wang, and Anders P. Ravn. 1995. A Formal Description of Hybrid Systems. In *Hybrid Systems III: Verification and Control (Lecture Notes in Computer Science, Vol. 1066)*. Springer, 511–530. doi:10.1007/BFb0020972

[395] Li Zhou, Gilles Barthe, Pierre-Yves Strub, Junyi Liu, and Mingsheng Ying. 2023. CoqQ: Foundational Verification of Quantum Programs. *Proc. ACM Program. Lang.* 7, POPL (2023), 833–865. doi:10.1145/3571222

[396] Li Zhou, Gilles Barthe, Pierre-Yves Strub, Junyi Liu, and Mingsheng Ying. 2023. CoqQ: Foundational Verification of Quantum Programs. *Proc. ACM Program. Lang.* 7, POPL (2023), 833–865. doi:10.1145/3571222

[397] Litao Zhou, Jianxing Qin, Qinshi Wang, Andrew W. Appel, and Qinxiang Cao. 2024. VST-A: A Foundationally Sound Annotation Verifier. *Proc. ACM Program. Lang.* 8, POPL (2024), 2069–2098. doi:10.1145/3632911

[398] Li Zhou, Nengkun Yu, and Mingsheng Ying. 2019. An applied quantum Hoare logic. In *PLDI 2019*. ACM, 1149–1162. doi:10.1145/3314221.3314584

[399] Lingfeng Zhu, Xindi Zhang, Yongjian Li, and Shaowei Cai. 2025. Leveraging Critical Proof Obligations for Efficient IC3 Verification. In *DAC 2025*. IEEE, 1–7. doi:10.1109/DAC63849.2025.11132734

[400] Liang Zou, Martin Fränzle, Naijun Zhan, and Peter Nazier Mosaad. 2015. Automatic Verification of Stability and Safety for Delay Differential Equations. In *CAV 2015 (Lecture Notes in Computer Science, Vol. 9207)*. Springer, 338–355. doi:10.1007/978-3-319-21668-3_20

[401] Liang Zou, Jidong Lv, Shuling Wang, Naijun Zhan, Tao Tang, Lei Yuan, and Yu Liu. 2013. Verifying Chinese Train Control System under a Combined Scenario by Theorem Proving. In *VSTTE 2013 (Lecture Notes in Computer Science, Vol. 8164)*. Springer, 262–280. doi:10.1007/978-3-642-54108-7_14

[402] Liang Zou, Naijun Zhan, Shuling Wang, and Martin Fränzle. 2015. Formal Verification of Simulink/Stateflow Diagrams. In *ATVA 2015 (Lecture Notes in Computer Science, Vol. 9364)*. Springer, 464–481. doi:10.1007/978-3-319-47016-0

[403] Liang Zou, Naijun Zhan, Shuling Wang, Martin Fränzle, and Shengchao Qin. 2013. Verifying Simulink diagrams via a Hybrid Hoare Logic Prover. In *EMSOFT 2013*. IEEE, 9:1–9:10. doi:10.1109/EMSOFT.2013.6658587

[404] Mo Zou, Haoran Ding, Dong Du, Ming Fu, Ronghui Gu, and Haibo Chen. 2019. Using concurrent relational logic with helpers for verifying the AtomFS file system. In *SOSP 2019*. ACM, 259–274. doi:10.1145/3341301.3359644

[405] Mo Zou, Dong Du, Mingkai Dong, and Haibo Chen. 2024. Using Dynamically Layered Definite Releases for Verifying the RefFS File System. In *OSDI 2024*. USENIX Association, 629–648. https://www.usenix.org/conference/osdi24/presentation/zou

[406] Zhiqiang Zuo, Rong Gu, Xi Jiang, Zhaokang Wang, Yihua Huang, Linzhang Wang, and Xuandong Li. 2019. BigSpa: An Efficient Interprocedural Static Analysis Engine in the Cloud. In *IPDPS 2019*. IEEE, 771–780. doi:10.1109/IPDPS.2019.00086

[407] Zhiqiang Zuo, John Thorpe, Yifei Wang, Qiuhong Pan, Shenming Lu, Kai Wang, Guoqing Harry Xu, Linzhang Wang, and Xuandong Li. 2019. Grapple: A Graph System for Static Finite-State Property Checking of Large-Scale Systems Code. In *EuroSys 2019*. ACM, 38:1–38:17. doi:10.1145/3302424.3303972

[408] Zhiqiang Zuo, Kai Wang, Aftab Hussain, Ardalan Amiri Sani, Yiyu Zhang, Shenming Lu, Wensheng Dou, Linzhang Wang, Xuandong Li, Chenxi Wang, and Guoqing Harry Xu. 2020. Systemizing Interprocedural Static Analysis of Large-scale Systems Code with Graspan. *ACM Trans. Comput. Syst.* 38, 1–2 (2020), 4:1–4:39. doi:10.1145/3466820

[409] Zhiqiang Zuo, Yiyu Zhang, Qiuhong Pan, Shenming Lu, Yue Li, Linzhang Wang, Xuandong Li, and Guoqing Harry Xu. 2021. Chianina: An evolving graph system for flow- and context-sensitive analyses of million lines of C code. In *PLDI 2021*. ACM, 914–929. doi:10.1145/3453483.3454085

Received 26 August 2025; revised 4 December 2025; accepted 8 December 2025