Cheng Zhang

Research Statement

czhang03@bu.edu
czhang03.github.io
czhang03
Curriculum vitae

"the possibility of making explicit, elegant computations has always come out [...] as a byproduct of a thorough conceptual understanding..." – Alexander Grothendieck

Many challenges in computer science can be formulated as program equivalences, where we seek to establish that two programs behave similarly or identically. For example, the correctness of compilers and decompilers can be defined by the equivalence of the input and output programs. Similarly, the correctness of machine-learning model compression can be verified by the equivalence of the model's behavior before and after compression. The integrity of privacy and security algorithms can also be characterized by their indistinguishability from ideal scenarios. More surprisingly, problems like network reachability, network traffic isolation, program logic, and computational effects, can all be formalized and analyzed using equations [1, 2, 3, 4, 5].

While general program equivalences remain undecidable, Kleene Algebra (KA), an abstract algebraic system rooted in automata theory, has shown great promise in both proving and automating program equivalences in various domains. The theoretical elegance of Kleene Algebra have inspired a wealth of foundational research [6, 7, 8, 9, 10, 11]; and the practical systems based on Kleene Algebra have excelled in compiler verification [12], (probabilistic) network systems [1, 13, 14, 15], weak memory model [16], and distributed systems [17]. Notably, some of these tools have matched or even outperformed state-of-the-art tools in their respective domains.

My research is nicely positioned at the intersection of theory and practice of Kleene Algebra, enabling me to collaborate with researchers and students with diverse backgrounds, and produce interesting results on both sides. My research workflow usually involves learning about specific application domains, applying the perspective of Kleene Algebra to understand and tackle practical problems, then refining theory to achieve the most elegant solutions. This approach has proven to be effective for me, yielding both rich theoretical frameworks and efficient tools, while also identifying promising open problems for future exploration. My current work has been applied to program logics [2, 18] and decompiler validation [19]; all published in top-tier conferences like POPL and ICALP. Additionally, my theoretical ventures have demonstrated the limitations of alignment reasoning in relational verification, resolved a long-standing open problem in Kleene Algebra [20], developed an algebraic perspective of the reduction technique in Kleene Algebra [18], and developed efficient algorithm to check the equivalence of uninterpreted programs [21].

Past and Ongoing Works

TopKAT: A Unified View Of Program Logic

Incorrectness logic [22], although simple, has shown great potential for bug detection across various semantic domains [23, 24, 25]. To unify the theory of incorrectness logic in these domains, we tried to use Kleene Algebra with Tests (KAT) as an algebraic semantic foundation. This task was soon proven to be impossible [2]: we showed that the theory of KAT is insufficient to encode incorrectness logic, mainly because KAT lacks the relational domain operation. Surprisingly, instead of adding a fully-fledged domain operator, we only need to add a top element. We named our framework Kleen algebra with top and tests (TopKAT), and unlike KAT extensions with domain operators [26, 27], TopKAT preserves the complexity

class of KAT [2].

However, upon diving into the theory of TopKAT, we discovered its unexpected limitation: despite its power to subsume both propositional Hoare and incorrectness logic, TopKAT is incomplete with respect to its relational model. Our followup work [18] resolves this weakness by focusing on the inequalities used to encode incorrectness and Hoare logic, which we named "domain-comparison inequalities". In this work, we used techniques in universal algebra to streamline the definition of reduction [28, 29]. This new perspective greatly simplified previous completeness proofs, and also allowed us to prove the relational completeness with regard domain-comparison inequalities. This result has not only demonstrated the effectiveness of reasoning about incorrectness and Hoare logics using TopKAT, but also other logics like reachability logic [30] as well.

CF-GKAT, control flow verification in nearly linear time

Control-flow manipulation is a prevalent task in software engineering, thus verifying its correctness is crucial to ensure software reliability. Our work, building upon foundational researches on Guarded Kleene Algebra with Tests (GKAT) [6] and the theory of non-local control flow in Kleene Algebra [31], greatly simplifies process of control-flow verification. Specifically, We extended GKAT automaton to handle common control structures, including break, return, goto, and indicator variables; while preserving its efficiency and correctness. These extensions enable us to validate a large class of control-flow restructuring algorithms [12, 32, 33, 34]. And its efficiency and correctness allow our works to be invoked automatically on-the-fly, or be used as a framework in a proof assistant.

Efficient Symbolic On-the-fly Algorithm for GKAT

In the process of implementing CF-GKAT, we have identified several ways to improve the efficiency of GKAT equivalences. For example, when there is a large amount of primitive test (primitive conditional statements used in if-statement and while-loops), the memory usage and runtime of the original algorithm [6] will blowup exponentially. The large memory usage is typically resolved using derivatives to produce the automaton on-the-fly [7, 35], whereas the long runtime can be optimized using symbolic automaton [36].

In collaboration with two undergrad students, our latest work marries these two ideas, and built a theory of symbolic GKAT coalgebra, which gives rises to several efficient symbolic equivalence-checking algorithms for GKAT. Unlike similar works on KAT [36], the structure of GKAT enables us to export the complex boolean logic into a fast and reliable solvers like z3; further improving the efficiency of our implementation. Our rust implementation can decide large equivalences (with more than 500 commands pre expression) in seconds with only couple megabytes of memory usage. This work also characterized the exact complexity of GKAT, which is co-NP-complete.

Kleene Algebra With Commutativity Hypothesis

Commutativity hypotheses have long been recognized for its importance in control-flow analysis [37], and recent work [38] has also established its vital role in relational verification. Contrary to its broad applications, the theory of KA with commutativity hypotheses remains stale; specifically, the decidability of the theory has made no progress since the question was raised by Kozen [37]. Independently, Kuznetsov [39] has shown that Kleene Algebra with commutativity is indeed undecidable. We, on the other hand, has shown the same result without using the induction or right unfolding rule [40]. Our result exhibits a large class of equational theories that are all undecidable when extended commutativity hypotheses, generalizing the result of Kuznetsov.

Vision for Future Work

My work has underscored the vast potential of algebraic methods in program verification, an approach that has gained significant attention recently as researchers seek to leverage the wealth of mathematical theory in real-world applications. For my future work, I am committed to continue bringing foundational research into the real-world, and discovering new interesting theoretical directions thorough practice.

Besides refining my current work and incorporating more practical features into systems like TopKAT and CF-GKAT, I also aim to expand the theory of algebraic reasoning into exciting new domains. One such domain is distributed systems and concurrency, where, despite a substantial body of semantic work [41, 42, 43, 44, 45], there is a notable gap between algebraic foundations and the developments of practical tools.

Two significant problems that plague concurrent programming and distributed systems are deadlocks and data-races. Although various tools have been designed to detect these undesirable behaviors [46, 47, 48, 49], many of these tools are language-specific and lack connections to existing semantic works. I aim to study these problems from an algebraic perspective based on well-developed semantic foundation, with the goal of developing both generic proof systems and efficient automation. Specifically, I believe there are two practical approaches which could benefit from said algebraic approach.

Session Types With Refinement Branching

Session types is a type system that specifies and verifies message-passing concurrency between two parties. While session type systems completely eliminate problems like deadlocks and data races [50, 51], they also impose restrictions on the types of programs that programmers can write. For instance, in leader election protocols, each server communicates the ID of the elected leader to its neighbor and select a protocol to execute depends on whether its ID coincide with the leader's. Traditional session types only allow protocol selection to depend on boolean values, which means that communicating the ID of the elected leader would require a stream of binary-valued packets to be reflected in the type annotations. This approach is not only computationally inefficient but also requires burdensome type for the programmer to annotate. More expressive versions of session types, such as dependent session types [52, 53] can resolve this problem, but their type equality and type checking are often undecidable.

Our work aims to find a reasonable middle ground by leveraging automaton theory and Kleene Algebra. We envision a type system that supports branching on communicated values, such as the leader's ID and the server's assigned ID, while also provide a robust, efficient, and semantics-based type equality checking algorithm. This algebraic approach would significantly improve the expressivity of traditional session type systems while preserving the efficiency of its type checking algorithm.

Deadlock and Data-Race Detection

In addition to type-based approaches to eliminating deadlocks and data races, I am also interested in detecting these problems in existing code bases without manual type annotations, particularly in programs with threads accessing shared memory. Despite the strong semantic foundation provided by Concurrent Kleene Algebra (CKA), current extensions of CKA lack two important features that are crucial for detecting deadlocks and data races in real-world programs. Firstly, they do not support the `future` construct, a popular concurrent structure used in many real-world languages. Secondly, they lack data dependency and memory models, which are essential in defining deadlocks and data races.

Augmenting these features to Concurrent Kleene Algebra will provide an algebraic understanding of real-world concurrent programs. This work will lead to the development of a proof system that can prove deadlock and data race freedom for a generic class of programs, as well as an automated checker for detecting deadlocks and data races. Furthermore, it may also enable a fast graph generation algorithm for

imperative programs like in graph types [54], and a deeper understanding of the connection between graph types and semantics.

Compositional Reduction Framework

However, all of these applications require significant extensions to existing systems of Kleene Algebra, and demonstrating that these extensions preserve the desirable properties of Kleene Algebra is a non-trivial task. This challenge has inspired my theoretical research to develop a compositional technique for proving important properties of extensions of Kleene Algebra.

Specifically, I envision a framework where we can prove theorems for each individual language feature, such as non-local control-flow, indicator variables, and top, and then combine them to derive a decision procedure and desirable meta-theorems without requiring additional proof. This result will not only be mathematically interesting but also save researchers a significant amount of time and effort when developing practical tools. In the future, we can even build formal libraries that allow users to select language features, and automatically generate the decision procedure with surrounding theorems. All of these visions are grounded in the algebraic notion of reduction that we discovered for TopKAT. We have recently found that this way of thinking is applicable to many existing systems, and its algebraic nature can potentially facilitate a compositional framework for combining language features.

In conclusion, my research highlights the broad applicability of algebraic reasoning in realistic problems of computer science. I see tremendous potential in the interplay between practicality and theory, where I apply foundational techniques to realistic problems and, in turn, identify new theoretical opportunities from the experience of solving practical issues. I believe that this workflow can lead to not only interesting mathematics and efficient tools but also long-running projects that can engage students and researchers from diverse backgrounds.

References

[1] Carolyn Jane Anderson et al. "NetKAT: Semantic Foundations for Networks". In: *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. San Diego California USA: ACM, Jan. 2014, pp. 113–126. ISBN: 978-1-4503-2544-8. DOI: 10.1145/2535838.2535862. (Visited on 12/08/2023).

[2] Cheng Zhang, Arthur Azevedo de Amorim, and Marco Gaboardi. "On Incorrectness Logic and Kleene Algebra with Top and Tests". In: *Proceedings of the ACM on Programming Languages* 6.POPL (Jan. 2022), 29:1–29:30. DOI: 10.1145/3498690.

[3] Bernhard Möller, Peter O'Hearn, and Tony Hoare. "On Algebra of Program Correctness and Incorrectness". In: *Relational and Algebraic Methods in Computer Science*. Ed. by Uli Fahrenberg et al. Vol. 13027. Cham: Springer International Publishing, 2021, pp. 325–343. ISBN: 978-3-030-88700-1 978-3-030-88701-8. DOI: 10.1007/978-3-030-88701-8_20. (Visited on 10/28/2021).

[4] Gordon Plotkin and Matija Pretnar. "Handlers of Algebraic Effects". In: *Programming Languages and Systems*. Ed. by Giuseppe Castagna. Vol. 5502. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 80–94. ISBN: 978-3-642-00589-3 978-3-642-00590-9. DOI: 10.1007/978-3-642-00590-9_7. (Visited on 11/03/2024).

[5] Gordon Plotkin and John Power. "Adequacy for Algebraic Effects". In: *Foundations of Software Science and Computation Structures*. Ed. by Gerhard Goos et al. Vol. 2030. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 1–24. ISBN: 978-3-540-41864-1 978-3-540-45315-4. DOI: 10.1007/3-540-45315-6_1. (Visited on 11/03/2024).

[6] Steffen Smolka et al. "Guarded Kleene Algebra with Tests: Verification of Uninterpreted Programs in Nearly Linear Time". In: *Proceedings of the ACM on Programming Languages* 4.POPL (Jan. 2020), pp. 1–28. ISSN: 2475-1421. DOI: 10.1145/3371129.

[7] Todd Schmid et al. *Guarded Kleene Algebra with Tests: Coequations, Coinduction, and Completeness*. May 2021. DOI: 10.4230/LIPIcs.ICALP.2021.142. arXiv: 2102.08286 [cs].

[8] Bart Jacobs. "A Bialgebraic Review of Deterministic Automata, Regular Expressions and Languages". In: *Algebra, Meaning, and Computation*. Ed. by David Hutchison et al. Vol. 4060. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 375–404. ISBN: 978-3-540-35462-8 978-3-540-35464-2. DOI: 10.1007/11780274_20. (Visited on 12/28/2023).

[9] Dexter Kozen. "On the Coalgebraic Theory of Kleene Algebra with Tests". In: (Mar. 2008). (Visited on 09/12/2022).

[10] Todd Junior Wayne Schmid. "Coalgebraic Completeness Theorems for Effectful Process Algebras". Doctoral. UCL (University College London), Jan. 2024. (Visited on 11/04/2024).

[11] Wojciech Różowski et al. *Probabilistic Guarded KAT Modulo Bisimilarity: Completeness and Complexity*. May 2023. DOI: 10.48550/arXiv.2305.01755. arXiv: 2305.01755. (Visited on 11/04/2024).

[12] Dexter Kozen and Maria-Christina Patron. "Certification of Compiler Optimizations Using Kleene Algebra with Tests". In: *Proceedings of the First International Conference on Computational Logic*. CL '00. Berlin, Heidelberg: Springer-Verlag, July 2000, pp. 568–582. ISBN: 978-3-540-67797-0. (Visited on 11/14/2024).

[13] Nate Foster et al. "A Coalgebraic Decision Procedure for NetKAT". In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '15. New York, NY, USA: Association for Computing Machinery, Jan. 2015, pp. 343–355. ISBN: 978-1-4503-3300-9. DOI: 10.1145/2676726.2677011. (Visited on 07/28/2023).

[14] Steffen Smolka et al. "Scalable Verification of Probabilistic Networks". In: *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. Phoenix AZ USA: ACM, June 2019, pp. 190–203. ISBN: 978-1-4503-6712-7. DOI: 10.1145/3314221.3314639. (Visited on 08/31/2021).

[15] Han Zhang et al. "Netter: Probabilistic, Stateful Network Models". In: *Verification, Model Checking, and Abstract Interpretation*. Ed. by Fritz Henglein, Sharon Shoham, and Yakir Vizel. Vol. 12597. Cham: Springer International Publishing, 2021, pp. 486–508. ISBN: 978-3-030-67066-5 978-3-030-67067-2. DOI: 10.1007/978-3-030-67067-2_22. (Visited on 08/31/2021).

[16] Michalis Kokologiannakis, Ori Lahav, and Viktor Vafeiadis. "Kater: Automating Weak Memory Model Metatheory and Consistency Checking". In: *Proceedings of the ACM on Programming Languages* 7.POPL (Jan. 2023), pp. 544–572. ISSN: 2475-1421. DOI: 10.1145/3571212. (Visited on 11/14/2024).

[17] A. K. McIver et al. "Using Probabilistic Kleene Algebra pKA for Protocol Verification". In: *The Journal of Logic and Algebraic Programming*. Relations and Kleene Algebras in Computer Science 76.1 (May 2008), pp. 90–111. ISSN: 1567-8326. DOI: 10.1016/j.jlap.2007.10.005. (Visited on 02/25/2022).

[18] Cheng Zhang, Arthur Azevedo de Amorim, and Marco Gaboardi. *Domain Reasoning in TopKAT*. Apr. 2024. DOI: 10.4230/LIPIcs.ICALP.2024.133. arXiv: 2404.18417 [cs].

[19] Cheng Zhang et al. "CF-GKAT: Efficient Validation of Control-Flow Transformations". In: *Proceedings of the ACM on Programming Languages* POPL (2025).

[20] Arthur Azevedo de Amorim, Cheng Zhang, and Marco Gaboardi. "Kleene Algebra with Commutativity Conditions Is Undecidable". Apr. 2024. (Visited on 11/14/2024).

[21] Cheng Zhang et al. *Efficient Symbolic Algorithms For GKAT Equivalence*. 2025.

[22] Peter W. O'Hearn. "Incorrectness Logic". In: *Proceedings of the ACM on Programming Languages* 4.POPL (Jan. 2020), pp. 1–32. ISSN: 2475-1421, 2475-1421. DOI: 10.1145/3371078.

[23] Azalea Raad et al. "Local Reasoning About the Presence of Bugs: Incorrectness Separation Logic". In: *Computer Aided Verification*. Ed. by Shuvendu K. Lahiri and Chao Wang. Vol. 12225. Cham: Springer International Publishing, 2020, pp. 225–252. ISBN: 978-3-030-53290-1 978-3-030-53291-8. DOI: 10.1007/978-3-030-53291-8_14.

[24] Quang Loc Le et al. "Finding Real Bugs in Big Programs with Incorrectness Logic". In: *Proceedings of the ACM on Programming Languages* 6.OOPSLA1 (Apr. 2022), pp. 1–27. ISSN: 2475-1421. DOI: 10.1145/3527325.

[25] Linpeng Zhang and Benjamin Lucien Kaminski. "Quantitative Strongest Post: A Calculus for Reasoning about the Flow of Quantitative Information". In: *Proceedings of the ACM on Programming Languages* 6.OOPSLA1 (Apr. 2022), 87:1–87:29. DOI: 10.1145/3527331.

[26] Jules Desharnais, Bernhard Möller, and Georg Struth. "Kleene Algebra with Domain". In: *ACM Transactions on Computational Logic* 7.4 (Oct. 2006), pp. 798–833. ISSN: 1529-3785. DOI: 10.1145/1183278.1183285.

[27] Igor Sedlár. "On the Complexity of Kleene Algebra with Domain". In: *Relational and Algebraic Methods in Computer Science: 20th International Conference, RAMiCS 2023, Augsburg, Germany, April 3–6, 2023, Proceedings*. Berlin, Heidelberg: Springer-Verlag, Apr. 2023, pp. 208–223. ISBN: 978-3-031-28082-5. DOI: 10.1007/978-3-031-28083-2_13.

[28] Damien Pous, Jurriaan Rot, and Jana Wagemaker. "On Tools for Completeness of Kleene Algebra with Hypotheses". In: *Relational and Algebraic Methods in Computer Science: 19th International Conference, RAMiCS 2021, Marseille, France, November 2–5, 2021, Proceedings*. Berlin, Heidelberg: Springer-Verlag, Nov. 2021, pp. 378–395. ISBN: 978-3-030-88700-1. DOI: 10.1007/978-3-030-88701-8_23.

[29] Dexter Kozen and Frederick Smith. "Kleene Algebra with Tests: Completeness and Decidability". In: *Computer Science Logic*. Ed. by Gerhard Goos et al. Vol. 1258. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 244–259. ISBN: 978-3-540-63172-9 978-3-540-69201-0. DOI: 10.1007/3-540-63172-0_43.

[30] Nico Naus et al. *Reachability Logic for Low-Level Programs*. Mar. 2022. DOI: 10.48550/arXiv.2204.00076. arXiv: 2204.00076 [cs].

[31] Dexter Kozen. "Nonlocal Flow of Control and Kleene Algebra with Tests". In: *2008 23rd Annual IEEE Symposium on Logic in Computer Science* (June 2008), pp. 105–117. ISSN: 1043-6871. DOI: 10.1109/LICS.2008.32.

[32] Khaled Yakdan et al. "No More Gotos: Decompilation Using Pattern-Independent Control-Flow Structuring and Semantics-Preserving Transformations". In: *Proceedings 2015 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, 2015. ISBN: 978-1-891562-38-9. DOI: 10.14722/ndss.2015.23185.

[33] Zion Leonahenahe Basque et al. "Ahoy SAILR! There Is No Need to DREAM of C: A Compiler-Aware Structuring Algorithm for Binary Decompilation". In: ().

[34] A.M. Erosa and L.J. Hendren. "Taming Control Flow: A Structured Approach to Eliminating Goto Statements". In: *Proceedings of 1994 IEEE International Conference on Computer Languages (ICCL'94)*. May 1994, pp. 229–240. DOI: 10.1109/ICCL.1994.288377.

[35] Janusz A. Brzozowski. "Derivatives of Regular Expressions". In: *Journal of the ACM* 11.4 (Oct. 1964), pp. 481–494. ISSN: 0004-5411. DOI: 10.1145/321239.321249.

[36] Damien Pous. "Symbolic Algorithms for Language Equivalence and Kleene Algebra with Tests". In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '15. New York, NY, USA: Association for Computing Machinery, Jan. 2015, pp. 357–368. ISBN: 978-1-4503-3300-9. DOI: 10.1145/2676726.2677007.

[37] Dexter Kozen. "Kleene Algebra with Tests and Commutativity Conditions". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by Gerhard Goos et al. Vol. 1055. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 14–33. ISBN: 978-3-540-61042-7 978-3-540-49874-2. DOI: 10.1007/3-540-61042-1_35.

[38] Timos Antonopoulos et al. "An Algebra of Alignment for Relational Verification". In: *Proceedings of the ACM on Programming Languages* 7.POPL (Jan. 2023), 20:573–20:603. DOI: 10.1145/3571213.

[39] Stepan L. Kuznetsov. "On the Complexity of Reasoning in Kleene Algebra with Commutativity Conditions". In: *Theoretical Aspects of Computing – ICTAC 2023*. Ed. by Erika Ábrahám, Clemens Dubslaff, and Silvia Lizeth Tapia Tarifa. Cham: Springer Nature Switzerland, 2023, pp. 83–99. ISBN: 978-3-031-47963-2. DOI: 10.1007/978-3-031-47963-2_7.

[40] Arthur Azevedo de Amorim, Cheng Zhang, and Marco Gaboardi. "Kleene Algebra with Commutativity Conditions Is Undecidable". Apr. 2024.

[41] Tony Hoare et al. "Concurrent Kleene Algebra and Its Foundations". In: *The Journal of Logic and Algebraic Programming*. Relations and Kleene Algebras in Computer Science 80.6 (Aug. 2011), pp. 266–296. ISSN: 1567-8326. DOI: 10.1016/j.jlap.2011.04.005. (Visited on 10/08/2021).

[42] Tobias Kappé et al. "Concurrent Kleene Algebra with Observations: From Hypotheses to Completeness". In: *Foundations of Software Science and Computation Structures*. Ed. by Jean Goubault-Larrecq and Barbara König. Vol. 12077. Cham: Springer International Publishing, 2020, pp. 381–400. ISBN: 978-3-030-45230-8 978-3-030-45231-5. DOI: 10.1007/978-3-030-45231-5_20. (Visited on 07/18/2023).

[43] Tobias Kappé et al. "Concurrent Kleene Algebra: Free Model and Completeness". In: *Programming Languages and Systems*. Ed. by Amal Ahmed. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 856–882. ISBN: 978-3-319-89884-1. DOI: 10.1007/978-3-319-89884-1_30.

[44] Jana Wagemaker et al. "Partially Observable Concurrent Kleene Algebra". In: *LIPIcs, Volume 171, CONCUR 2020* 171 (2020), 20:1–20:22. ISSN: 1868-8969. DOI: 10.4230/LIPIcs.CONCUR.2020.20. arXiv: 2007.07593 [cs]. (Visited on 11/08/2024).

[45] Annabelle McIver, Tahiry Rabehaja, and Georg Struth. "Probabilistic Concurrent Kleene Algebra". In: *Electronic Proceedings in Theoretical Computer Science* 117 (June 2013), pp. 97–115. ISSN: 2075-2180. DOI: 10.4204/EPTCS.117.7. (Visited on 02/03/2022).

[46] Amy Williams, William Thies, and Michael D. Ernst. "Static Deadlock Detection for Java Libraries". In: *ECOOP 2005 - Object-Oriented Programming*. Ed. by David Hutchison et al. Vol. 3586. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 602–629. ISBN: 978-3-540-27992-1 978-3-540-31725-8. DOI: 10.1007/11531142_26. (Visited on 11/10/2024).

[47] Stephen P. Masticola. "Static Detection of Deadlocks in Polynomial Time". In: (1993). DOI: 10.7282/T3-95QX-2391. (Visited on 11/10/2024).

[48] Dawson Engler and Ken Ashcraft. "RacerX: Effective, Static Detection of Race Conditions and Deadlocks". In: *SIGOPS Oper. Syst. Rev.* 37.5 (Oct. 2003), pp. 237–252. ISSN: 0163-5980. DOI: 10.1145/1165389.945468. (Visited on 11/10/2024).

[49] Mayur Naik et al. "Effective Static Deadlock Detection". In: *2009 IEEE 31st International Conference on Software Engineering*. Vancouver, BC, Canada: IEEE, 2009, pp. 386–396. ISBN: 978-1-4244-3453-4. DOI: 10.1109/ICSE.2009.5070538. (Visited on 11/10/2024).

[50] Luís Caires and Frank Pfenning. "Session Types as Intuitionistic Linear Propositions". In: *CONCUR 2010 - Concurrency Theory*. Ed. by Paul Gastin and François Laroussinie. Berlin, Heidelberg: Springer, 2010, pp. 222–236. ISBN: 978-3-642-15375-4. DOI: 10.1007/978-3-642-15375-4_16.

[51] Philip Wadler. "Propositions as Sessions". In: *SIGPLAN Not.* 47.9 (Sept. 2012), pp. 273–286. ISSN: 0362-1340. DOI: 10.1145/2398856.2364568. (Visited on 11/22/2024).

[52] Bernardo Toninho, Luís Caires, and Frank Pfenning. "Dependent Session Types via Intuitionistic Linear Type Theory". In: *Proceedings of the 13th International ACM SIGPLAN Symposium on Principles and Practices of Declarative Programming*. PPDP '11. New York, NY, USA: Association for Computing Machinery, July 2011, pp. 161–172. ISBN: 978-1-4503-0776-5. DOI: 10.1145/2003476.2003499. (Visited on 11/19/2024).

[53] Bernardo Toninho, Luís Caires, and Frank Pfenning. "A Decade of Dependent Session Types". In: *23rd International Symposium on Principles and Practice of Declarative Programming*. Tallinn Estonia: ACM, Sept. 2021, pp. 1–3. ISBN: 978-1-4503-8689-0. DOI: 10.1145/3479394.3479398. (Visited on 11/19/2024).

[54] Stefan K Muller. "Language-Agnostic Static Deadlock Detection for Futures". In: *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*. Edinburgh United Kingdom: ACM, Mar. 2024, pp. 68–79. ISBN: 9798400704352. DOI: 10.1145/3627535.3638487. (Visited on 11/10/2024).