Formal Methods Review

Manuscript Formal Methods Review

2019-02-08 21:37

Richard L. Ford
richardlford@gmail.com

Abstract

This is a review of current formal methods theory, tools, projects and people.

Contents

0. Introduction 1
1. References 1
References 1

0. Introduction

Bi-Abduction is define in (Calcagno et al., 2011).

1. References

References

[Absint, n.d.] Absint. (n.d.). CompCert - Publications. Retrieved January 31, 2019, from http://compcert.inria.fr/publi.html

Adams, 2015] Adams, M. (2015). The Common HOL Platform. *Electronic Proceedings in Theoretical Computer Science*, *186*, 42–56. https://doi.org/10.4204/EPTCS.186.6

vale, n.d.] Adewale, O. (n.d.). Implementing a high-performance key-value store using a trie of B+-Trees with cursors | Computer Science Department at Princeton University. Retrieved February 1, 2019, from https://www.cs.princeton.edu/research/techreps/TR-004-18

al., 2017] Ahman, D., Fournet, C., Hriţcu, C., Maillard, K., Rastogi, A., & Swamy, N. (2017). Recalling a Witness: Foundations and Applications of Monotonic State. *Proc. ACM Program. Lang.*, *2*, 65:1–65:30. https://doi.org/10.1145/3158153

al., 2017] Ahman, D., Hriţcu, C., Maillard, K., Martínez, G., Plotkin, G., Protzenko, J., … Swamy, N. (2017). Dijkstra Monads for Free. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages* (pp. 515–529). New York, NY, USA: ACM. https://doi.org/10.1145/3009837.3009878

ndt, n.d.] Ahrendt, W. (n.d.). Deductive Software Verification – The KeY BookFrom Theory to Practice – The KeY Project. Retrieved January 31, 2019, from https://www.key-project.org/thebook2/

itt, 2007] Ahrendt, W., Beckert, B., Hähnle, R., Rümmer, P., & Schmitt, P. H. (2007). Verifying Object-Oriented Programs with KeY: A Tutorial. In F. S. de Boer, M. M. Bonsangue, S. Graf, & W.-P. de Roever (Eds.), *Formal Methods for Components and Objects* (Vol. 4709, pp. 70–101). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-74792-5_4

erg, 2018] Altenkirch, T., Capriotti, P., Dijkstra, G., Kraus, N., & Forsberg, F. N. (2018). Quotient inductive-inductive types. *arXiv:1612.02346 [cs]*, *10803*, 293–310. https://doi.org/10.1007/978-3-319-89366-2_16

npf, 2016] Amin, N., Leino, R., & Rompf, T. (2016). Computing with an SMT Solver, *8570*. Retrieved from https://www.microsoft.com/en-us/research/publication/computing-smt-solver/

al., 2013] Amorim, A. A. de, Collins, N., DeHon, A., Demange, D., Hritcu, C., Pichardie, D., … Tolmach, A. (2013). *A Verified Information-Flow Architecture (Long version)*.

eau, n.d.] Anand, A., Boulier, S., Cohen, C., Sozeau, M., & Tabareau, N. (n.d.). Towards Certified Meta-Programming with Typed Template-Coq | SpringerLink. Retrieved February 1, 2019, from https://link.springer.com/chapter/10.1007%2F978-3-319-94821-8_2

eau, n.d.] Anand, A., Tabareau, S. B. N., & Sozeau, M. (n.d.). Typed Template Coq, 2.

ew, 2008] Andrew, A. (2008). *Oracle Semantics Aquinas Hobor*.

. et al., 2017] Appel Andrew W., Beringer Lennart, Chlipala Adam, Pierce Benjamin C., Shao Zhong, Weirich Stephanie, & Zdancewic Steve. (2017). Position paper: the science of deep specification. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *375*(2104), 20160331. https://doi.org/10.1098/rsta.2016.0331

[Appel, 2011] Appel, A. W. (2011). VeriSmall: Verified Smallfoot Shape Analysis. In J.-P. Jouannaud & Z. Shao (Eds.), *Certified Programs and Proofs* (Vol. 7086, pp. 231–246). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-25379-9_18

[Appel, 2012] Appel, A. W. (2012). Verified Software Toolchain. In *Proceedings of the 4th International Conference on NASA Formal Methods* (pp. 2–2). Berlin, Heidelberg: Springer-Verlag. https://doi.org/10.1007/978-3-642-28891-3_2

[Appel, 2015] Appel, A. W. (2015). Verification of a Cryptographic Primitive: SHA-256. *ACM Trans. Program. Lang. Syst.*, *37*(2), 7:1–7:31. https://doi.org/10.1145/2701415

[Appel, 2019] Appel, A. W. (2019). *DeepSpecDB - github*. PrincetonUniversity. Retrieved from https://github.com/PrincetonUniversity/DeepSpecDB

[Appel, n.d.] Appel, A. W. (n.d.). CertiCoq: A verified compiler for Coq - POPL 2017. Retrieved February 1, 2019, from https://popl17.sigplan.org/event/main-certicoq-a-verified-compiler-for-coq

l et al., 2014] Appel, A. W., Dockins, R., Hobor, A., Beringer, L., Dodds, J., Stewart, G., … Leroy, X. (2014). *Verifiabble C, Version 2.2*. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9781107256552

uvelot, 2017] Arias, E. J. G., Pin, B., & Jouvelot, P. (2017). jsCoq: Towards Hybrid Theorem Proving Interfaces. *Electronic Proceedings in Theoretical Computer Science*, *239*, 15–27. https://doi.org/10.4204/EPTCS.239.2

n et al., 2014] Azevedo de Amorim, A., Collins, N., DeHon, A., Demange, D., Hriţcu, C., Pichardie, D., … Tolmach, A. (2014). A Verified Information-flow Architecture. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (pp. 165–178). New York, NY, USA: ACM. https://doi.org/10.1145/2535838.2535839

t Appel, n.d.] Barriere, A., & Appel, A. (n.d.). VST Verification of B+Trees with Cursors, 19.

White, 1971] Bate, R. R., Mueller, D. D., & White, J. E. (1971). *Fundamentals of astrodynamics*. New York: Dover Publications.

ark, 2017] Batty Mark. (2017). Compositional relaxed concurrency. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *375*(2104), 20150406. https://doi.org/10.1098/rsta.2015.0406

itt, 2006] Beckert, B., Hähnle, R., & Schmitt, P. H. (Eds.). (2006). *Verification of Object-Oriented Software. The KeY Approach* (Vol. 4334). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-69061-0

ord, 2017] Bedford, A. (2017). Coqatoo: Generating Natural Language Versions of Coq Proofs. *arXiv:1712.03894 [cs]*. Retrieved from arXiv:1712.03894

ford, n.d.] Bedford, A. (n.d.). Coqatoo: Generating Natural Language Versions of Coq Proofs - Slides, 16.

arn, 2006] Berdine, J., Calcagno, C., & O'Hearn, P. W. (2006). Smallfoot: Modular Automatic Assertion Checking with Separation Logic. In F. S. de Boer, M. M. Bonsangue, S. Graf, & W.-P. de Roever (Eds.), *Formal Methods for Components and Objects* (pp. 115–137). Springer Berlin Heidelberg.

pel, 2014] Beringer, L., Stewart, G., Dockins, R., & Appel, A. W. (2014). Verified Compilation for Shared-Memory C. In Z. Shao (Ed.), *Programming Languages and Systems* (Vol. 8410, pp. 107–127). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-54833-8_7,

rtot, n.d.] Bertot, Y. (n.d.). Yves Bertot. Retrieved January 31, 2019, from http://www-sop.inria.fr/members/Yves.Bertot/index.html

ran, 2004] Bertot, Y., & Castéran, P. (2004). *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Berlin ; New York: Springer. Retrieved from http://www.labri.fr/perso/casteran/CoqArt/index.html

zjak, n.d.] Birkedal, L., & Bizjak, A. (n.d.). Iris Tutorial. Retrieved February 1, 2019, from https://iris-project.org/tutorial-material.html

tov, 2017] Blanchard, A., Loulergue, F., & Kosmatov, N. (2017). From Concurrent Programs to Simulating Sequential Programs: Correctness of a Transformation. *Electronic Proceedings in Theoretical Computer Science*, *253*, 109–123. https://doi.org/10.4204/EPTCS.253.9

iot, 2018] Blatter, L., Kosmatov, N., Le Gall, P., Prevosto, V., & Petiot, G. (2018). Static and Dynamic Verification of Relational Properties on Self-composed C Code. In C. Dubois & B. Wolff (Eds.), *Tests and Proofs* (pp. 44–62). Springer International Publishing.

Platzer, 2018] Bohrer, B., Tan, Y. K., Mitsch, S., Myreen, M. O., & Platzer, A. (2018). VeriPhy: verified controller executables from verified cyber-physical system models. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation - PLDI 2018* (pp. 617–630). Philadelphia, PA, USA: ACM Press. https://doi.org/10.1145/3192366.3192406

poutot, 2017] Boldo, S., Faissole, F., & Chapoutot, A. (2017). Round-off Error Analysis of Explicit One-Step Numerical Integration Methods. In *24th IEEE Symposium on Computer Arithmetic*. London, United Kingdom. https://doi.org/10.1109/ARITH.2017.22

poutot, 2018] Boldo, S., Faissole, F., & Chapoutot, A. (2018). *Round-off error and exceptional behavior analysis of explicit Runge-Kutta methods*. Retrieved from https://hal.archives-ouvertes.fr/hal-01883843

quiond, 2012] Boldo, S., Lelay, C., & Melquiond, G. (2012). Improving Real Analysis in Coq: A User-Friendly Approach to Integrals and Derivatives. In C. Hawblitzel & D. Miller (Eds.), *Certified Programs and Proofs* (Vol. 7679, pp. 289–304). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35308-6_22

quiond, 2013] Boldo, S., Lelay, C., & Melquiond, G. (2013). Coquelicot: A User-Friendly Library of Real Analysis for Coq. Retrieved from https://hal.inria.fr/hal-00860648/document

quiond, 2016] Boldo, S., Lelay, C., & Melquiond, G. (2016). Formalization of Real Analysis: A Survey of Proof Assistants and Libraries. *Mathematical Structures in Computer Science*, *26*(7), 1196–1233. https://doi.org/10.1017/S0960129514000437

bareau, 2017] Boulier, S., Pédrot, P.-M., & Tabareau, N. (2017). The next 700 syntactical models of type theory (pp. 182–194). https://doi.org/10.1145/3018610.3018620

owman, n.d.] Bowman, J. (n.d.). *J1: a small Forth CPU Core for FPGAs*.

i et al., 2018] Brahmi, A., Delmas, D., Essoussi, M. H., Randimbivololona, F., Atki, A., & Marie, T. (2018). Formalise to automate: deployment of a safe and cost-efficient process for avionics software. In *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*. Toulouse, France. Retrieved from https://hal.archives-ouvertes.fr/hal-01708332

hi et al., n.d.] Brahmi, A., Delmas, D., Essoussi, M. H., Randimbivololona, F., Informatics, C., Nauzere, L., … Marie, T. (n.d.). Formalise to automate: deployment of a safe and cost-efficient process for avionics software -Extended, 17.

erman, 2016] Brockschmidt, M., Cook, B., Ishtiaq, S., Khlaaf, H., & Piterman, N. (2016). T2: Temporal Property Verification. In M. Chechik & J.-F. Raskin (Eds.), *Tools and Algorithms for the Construction and Analysis of Systems* (pp. 387–393). Springer Berlin Heidelberg.

kes, 2007] Brookes, S. (2007). A semantics for concurrent separation logic. *Theoretical Computer Science*, *375*(1), 227–270. https://doi.org/10.1016/j.tcs.2006.12.034

arn, 2016] Brookes, S., & O'Hearn, P. W. (2016). Concurrent Separation Logic. *ACM SIGLOG News*, *3*(3), 47–65. https://doi.org/10.1145/2984450.2984457

earn, n.d.] Calcagno, C., Distefano, D., Dubreil, J., & O'Hearn, P. (n.d.). Moving Fast with Software Verification.Facebook Research. Retrieved February 1, 2019, from https://research.fb.com/publications/moving-fast-with-software-verification

ang, 2011] Calcagno, C., Distefano, D., O'Hearn, P. W., & Yang, H. (2011). Compositional Shape Analysis by Means of Bi-Abduction. *Journal of the ACM*, *58*(6), 1–66. https://doi.org/10.1145/2049697.2049700

pel, 2018] Cao, Q., Beringer, L., Gruetter, S., Dodds, J., & Appel, A. W. (2018). VST-Floyd: A Separation Logic Tool to Verify Correctness of C Programs. *J. Autom. Reason.*, *61*(1), 367–422. https://doi.org/10.1007/s10817-018-9457-5

éran, n.d.] Castéran, P. (n.d.). Pierre Castéran's Home page. Retrieved January 31, 2019, from http://www.labri.fr/perso/casteran/index.html

ud, 2010a] Charguéraud, A. (2010a). *Characteristic Formulae for Mechanized Program Verification* (phdthesis). UNIVERSITÉ PARIS.DIDEROT, Paris, France.

ud, 2010b] Charguéraud, A. (2010b). Program Verification Through Characteristic Formulae. In *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming* (pp. 321–332). New York, NY, USA: ACM. https://doi.org/10.1145/1863543.1863590

aud, 2011] Charguéraud, A. (2011). Characteristic Formulae for the Verification of Imperative Programs. In *Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming* (pp. 418–430). New York, NY, USA: ACM. https://doi.org/10.1145/2034773.2034828

hen, n.d.] Chen, Y. (n.d.). Project Report on DeepSpecDB, 35.

ala, 2013] Chlipala, A. (2013). *Certified programming with dependent types: a pragmatic introduction to the Coq proof assistant*. Cambridge, MA: The MIT Press. Retrieved from http://adam.chlipala.net/cpdt/

ala, 2019] Chlipala, A. (2019). *Formal Reasoning About Programs - Github*. Retrieved from https://github.com/achlipala/frap

la, n.d.-a] Chlipala, A. (n.d.-a). An Introduction to Programming and Proving with Dependent Types in Coq. *Journal of Formalized Reasoning*, *3*(2), 93.

ipala, n.d.-b] Chlipala, A. (n.d.-b). Certified Programming with Dependent Types, 369.

a et al., n.d.] Chlipala, A., Delaware, B., Duchovni, S., Gross, J., Pit-Claudel, C., Suriyakarn, S., … ye, K. (n.d.). THE END OF HISTORY? USING A PROOF ASSISTANT TO REPLACE LANGUAGE DESIGN WITH LIBRARY DESIGN. Retrieved February 1, 2019, from https://snapl.org/2017/abstracts/Chlipala.html

Arvind, 2017] Choi, J., Vijayaraghavan, M., Sherman, B., Chlipala, A., & Arvind. (2017). Kami: A Platform for High-level Parametric Hardware Specification and Its Modular Verification. *Proc. ACM Program. Lang.*, *1*, 24:1–24:30. https://doi.org/10.1145/3110268

üstholz, 2012] Christakis, M., Müller, P., & Wüstholz, V. (2012). Collaborative Verification and Testing with Explicit Assumptions. In D. Giannakopoulou & D. Méry (Eds.), *FM 2012: Formal Methods* (pp. 132–146). Springer Berlin Heidelberg.

ebsout, 2018] Conchon, S., Coquereau, A., Iguernlala, M., & Mebsout, A. (2018). Alt-Ergo 2.2. In *SMT Workshop: International Workshop on Satisfiability Modulo Theories*. Oxford, United Kingdom. Retrieved from https://hal.inria.fr/hal-01960203

ernlala, 2016] Conchon, S., & Iguernlala, M. (2016). Increasing Proofs Automation Rate of Atelier-B Thanks to Alt-Ergo. In T. Lecomte, R. Pinger, & A. Romanovsky (Eds.), *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification* (pp. 243–253). Springer International Publishing.

evadas, 2016] Costan, V., Lebedev, I., & Devadas, S. (2016). Sanctum: Minimal Hardware Extensions for Strong Software Isolation (pp. 857–874). Retrieved from https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/costan

vadas, 2017a] Costan, V., Lebedev, I., & Devadas, S. (2017a). Secure Processors Part I: Background, Taxonomy for Secure Enclaves and Intel SGX Architecture. *Foundations and Trends® in Electronic Design Automation*, *11*(1), 1–248. https://doi.org/10.1561/1000000051

vadas, 2017b] Costan, V., Lebedev, I., & Devadas, S. (2017b). Secure Processors Part II: Intel SGX Security Analysis and MIT Sanctum Architecture. *Foundations and Trends® in Electronic Design Automation*, *11*(3), 249–361. https://doi.org/10.1561/1000000052

& Gu, 2016] Costanzo, D., Shao, Z., & Gu, R. (2016). End-to-end Verification of Information-flow Security for C and Assembly Programs. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation* (pp. 648–664). New York, NY, USA: ACM. https://doi.org/10.1145/2908080.2908100

Gu, n.d.] Costanzo, D., Shao, Z., & Gu, R. (n.d.). End-to-End Verification of Information-Flow Security for C and Assembly Programs - Tech Report, 21. Retrieved from http://flint.cs.yale.edu/certikos/publications/security-tr.pdf

ary, 2017] Crary, K. (2017). Modules, Abstraction, and Parametric Polymorphism. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages* (pp. 100–113). New York, NY, USA: ACM. https://doi.org/10.1145/3009837.3009892

eda, 2014] Crick, T., Hall, B. A., Ishtiaq, S., & Takeda, K. (2014). "Share and Enjoy": Publishing Useful and Usable Scientific Models. *arXiv:1409.0367 [cs]*. Retrieved from arXiv:1409.0367

szyk, n.d.] Czajka, L., & Kaliszyk, C. (n.d.). CoqHammer: Strong Automation for Program Verification - CoqPL 2018. Retrieved January 31, 2019, from https://popl18.sigplan.org/event/coqpl-2018-coqhammer-strong-automation-for-program-verification

niel, 2017] David Cristina, & Kroening Daniel. (2017). Program synthesis: challenges and opportunities. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *375*(2104), 20150403. https://doi.org/10.1098/rsta.2015.0403

aye, 2000] Delahaye, D. (2000). A Tactic Language for the System Coq. In M. Parigot & A. Voronkov (Eds.), *Logic for Programming and Automated Reasoning* (Vol. 1955, pp. 85–95). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-44404-1_7

ala, 2015] Delaware, B., Pit-Claudel, C., Gross, J., & Chlipala, A. (2015). Fiat: Deductive Synthesis of Abstract Data Types in a Proof Assistant. In *Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (pp. 689–700). New York, NY, USA: ACM. https://doi.org/10.1145/2676726.2677006

pala, n.d.] Delaware, B., Suriyakarn, S., Pit-Claudel, C., Ye, Q., & Chlipala, A. (n.d.). Narcissus: Correct-By-Construction Derivation of Decoders and Encoders from Binary Formats, 14.

ala, 2018] Delaware, B., Suriyakarn, S., Pit–Claudel, C., Ye, Q., & Chlipala, A. (2018). Narcissus: Deriving Correct-By-Construction Decoders and Encoders from Binary Formats. Retrieved from https://arxiv.org/abs/1803.04870v2

al., 2017] Delignat-Lavaud, A., Fournet, C., Kohlweiss, M., Protzenko, J., Rastogi, A., Swamy, N., … Zinzindohoue, J. K. (2017). Implementing and Proving the TLS 1.3 Record Layer. Retrieved from https://www.microsoft.com/en-us/research/publication/implementing-proving-tls-1-3-record-layer/

tra, 1975] Dijkstra, E. W. (1975). Guarded Commands, Nondeterminacy and Formal Derivation of Programs. *Commun. ACM*, *18*(8), 453–457. https://doi.org/10.1145/360933.360975

t Yang, 2006] Distefano, D., O'Hearn, P. W., & Yang, H. (2006). A Local Shape Analysis Based on Separation Logic. In H. Hermanns & J. Palsberg (Eds.), *Tools and Algorithms for the Construction and Analysis of Systems* (Vol. 3920, pp. 287–302). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/11691372_19

Appel, 2009] Dockins, R., Hobor, A., & Appel, A. W. (2009). A Fresh Look at Separation Algebras and Share Accounting. In Z. Hu (Ed.), *Programming Languages and Systems* (Vol. 5904, pp. 161–177). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-10672-9_13

Moura, 2017] Ebner, G., Ullrich, S., Roesch, J., Avigad, J., & Moura, L. de. (2017). A Metaprogramming Framework for Formal Verification. *Proc. ACM Program. Lang.*, *1*, 34:1–34:29. https://doi.org/10.1145/3110278

i et al., 2017] Ekici, B., Mebsout, A., Tinelli, C., Keller, C., Katz, G., Reynolds, A., & Barrett, C. (2017). SMTCoq: A Plug-In for Integrating SMT Solvers into Coq. In R. Majumdar & V. Kunčak (Eds.), *Computer Aided Verification* (pp. 126–133). Springer International Publishing.

ilinski, 1994] Filinski, A. (1994). Representing Monads. In *Proceedings of the 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (pp. 446–457). New York, NY, USA: ACM. https://doi.org/10.1145/174675.178047

ilinski, 1999] Filinski, A. (1999). Representing Layered Monads. In *Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (pp. 175–188). New York, NY, USA: ACM. https://doi.org/10.1145/292540.292557

ymond, 2017] Fisher Kathleen, Launchbury John, & Richards Raymond. (2017). The HACMS program: using formal methods to eliminate exploitable bugs. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *375*(2104), 20150401. https://doi.org/10.1098/rsta.2015.0401

Swamy, n.d.] Fournet, C., Hawblitzel, C., Parno, B., & Swamy, N. (n.d.). Deploying a Verified Secure Implementation of the HTTPS Ecosystem, 10.

[Fowler, n.d.] Fowler, M. (n.d.). Deriving Kepler's Laws from the Inverse-Square Law. Retrieved February 1, 2019, from http://galileo.phys.virginia.edu/classes/152.mf1i.spring02/KeplersLaws.htm

Platzer, 2015] Fulton, N., Mitsch, S., Quesel, J.-D., Völp, M., & Platzer, A. (2015). KeYmaera X: An Axiomatic Tactical Theorem Prover for Hybrid Systems. In A. P. Felty & A. Middeldorp (Eds.), *Automated Deduction - CADE-25* (Vol. 9195, pp. 527–538). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-21401-6_36

ser, 1988] Gasser, M. (1988). *Building a secure computer system*. New York: Van Nostrand Reinhold Co.

ier, 2008] Gonthier, G. (2008). Formal Proof—The Four- Color Theorem, *55*(11), 12.

ubi, 2010] Gonthier, G., & Mahboubi, A. (2010). An introduction to small scale reflection in Coq. *Journal of Formalized Reasoning*, *3*(2), 95–152. https://doi.org/10.6092/issn.1972-5787/1979

yer, 2011] Gonthier, G., Ziliani, B., Nanevski, A., & Dreyer, D. (2011). How to Make Ad Hoc Proof Automation Less Ad Hoc. In *Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming* (pp. 163–175). New York, NY, USA: ACM. https://doi.org/10.1145/2034773.2034798

gey, 2019] Gorogiannis, N., O'Hearn, P. W., & Sergey, I. (2019). A true positives theorem for a static race detector. *Proceedings of the ACM on Programming Languages*, *3*, 1–29. https://doi.org/10.1145/3290370

al., 2015] Gu, R., Koenig, J., Ramananandro, T., Shao, Z., Wu, X. (Newman), Weng, S.-C., … Guo, Y. (2015). Deep Specifications and Certified Abstraction Layers. In *Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (pp. 595–608). New York, NY, USA: ACM. https://doi.org/10.1145/2676726.2676975

al., 2016] Gu, R., Shao, Z., Chen, H., Wu, X., Kim, J., Sjöberg, V., & Costanzo, D. (2016). CertiKOS: An Extensible Architecture for Building Certified Concurrent OS Kernels. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (pp. 653–669). Berkeley, CA, USA: USENIX Association. Retrieved from http://dl.acm.org/citation.cfm?id=3026877.3026928

al., 2018] Gu, R., Shao, Z., Kim, J., Wu, X. (Newman), Koenig, J., Sjöberg, V., … Ramananandro, T. (2018). Certified Concurrent Abstraction Layers. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation* (pp. 646–661). New York, NY, USA: ACM. https://doi.org/10.1145/3192366.3192381

son, 2013] Haase, C., Ishtiaq, S., Ouaknine, J., & Parkinson, M. J. (2013). SeLoger: A Tool for Graph-Based Reasoning in Separation Logic. In N. Sharygina & H. Veith (Eds.), *Computer Aided Verification* (Vol. 8044, pp. 790–795). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-39799-8_55

kin, 1993] Harper, R., Honsell, F., & Plotkin, G. (1993). A Framework for Defining Logics. *J. ACM*, *40*(1), 143–184. https://doi.org/10.1145/138027.138060

son, 2008] Harrison, J. (2008). Formal Proof—Theory and Practice, *55*(11), 12.

rrison, 2013] Harrison, J. (2013). The HOL Light Theory of Euclidean Space. *Journal of Automated Reasoning*, *50*(2), 173–190. https://doi.org/10.1007/s10817-012-9250-9

kinson, 2012] Hatcliff, J., Leavens, G. T., Leino, K. R. M., Müller, P., & Parkinson, M. (2012). Behavioral Interface Specification Languages. *ACM Comput. Surv.*, *44*(3), 16:1–16:58. https://doi.org/10.1145/2187671.2187678

Roşu, 2015] Hathhorn, C., Ellison, C., & Roşu, G. (2015). Defining the Undefinedness of C. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation* (pp. 336–345). New York, NY, USA: ACM. https://doi.org/10.1145/2737924.2737979

, et al., 2015] Hawblitzel, C., Howell, J., Kapritsos, M., Lorch, J. R., Parno, B., Roberts, M. L., … Zill, B. (2015). IronFleet: Proving Practical Distributed Systems Correct. In *Proceedings of the 25th Symposium on Operating Systems Principles* (pp. 1–17). New York, NY, USA: ACM. https://doi.org/10.1145/2815400.2815428

el et al., n.d.] Hawblitzel, C., Howell, J., Lorch, J. R., Narayan, A., Parno, B., Zhang, D., & Zill, B. (n.d.). Ironclad Apps: End-to-End Security via Automated Full-System Verification, 18.

Tasiran, 2015] Hawblitzel, C., Petrank, E., Qadeer, S., & Tasiran, S. (2015). Automated and Modular Refinement Reasoning for Concurrent Programs. In *Computer Aided Verification* (pp. 449–465). Springer, Cham. https://doi.org/10.1007/978-3-319-21668-3_26

Wing, 1990] Herlihy, M. P., & Wing, J. M. (1990). Linearizability: A Correctness Condition for Concurrent Objects. *ACM Trans. Program. Lang. Syst.*, *12*(3), 463–492. https://doi.org/10.1145/78969.78972

Appel, 2010] Hobor, A., Dockins, R., & Appel, A. W. (2010). A Theory of Indirection via Approximation. In *Proceedings of the 37th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (pp. 171–184). New York, NY, USA: ACM. https://doi.org/10.1145/1706299.1706322

uffman, 2013] Hölzl, J., Immler, F., & Huffman, B. (2013). Type Classes and Filters for Mathematical Analysis in Isabelle/HOL. In S. Blazy, C. Paulin-Mohring, & D. Pichardie (Eds.), *Interactive Theorem Proving* (pp. 279–294). Springer Berlin Heidelberg.

Hriţcu, 2015] Hriţcu, C. (2015). Micro-Policies: Formally Verified, Tag-Based Security Monitors. In *Proceedings of the 10th ACM Workshop on Programming Languages and Analysis for Security - PLAS'15* (pp. 1–1). Prague, Czech Republic: ACM Press. https://doi.org/10.1145/2786558.2786560

Hriţcu, n.d.] Hriţcu, C. (n.d.). The Quest for Formally Secure Compartmentalizing Compilation, 96.

ana, 2017] Hunt Warren A., Kaufmann Matt, Moore J Strother, & Slobodova Anna. (2017). Industrial hardware and software verification with ACL2. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *375*(2104), 20150399. https://doi.org/10.1098/rsta.2015.0399

ler, 2018] Immler, F. (2018). A Verified ODE Solver and the Lorenz Attractor. *J Autom Reasoning*, *61*(1), 73–111. https://doi.org/10.1007/s10817-017-9448-y

nria, n.d.] Inria. (n.d.). Inria - Inventors for the digital world.Inria. Retrieved January 31, 2019, from https://www.inria.fr/en

arn, 2011] Ishtiaq, S., & O'Hearn, P. W. (2011). BI As an Assertion Language for Mutable Data Structures. *SIGPLAN Not.*, *46*(4), 84–96. https://doi.org/10.1145/1988042.1988050

obs, 2019] Jacobs, B. (2019). *verifast/verifast: Research prototype tool for modular formal verification of C and Java programs*. verifast. Retrieved from https://github.com/verifast/verifast

ens, 2008] Jacobs, B., & Piessens, F. (2008). *The VeriFast program verifier*.

ens, 2017] Jacobs, B., Smans, J., & Piessens, F. (2017). The VeriFast Program Verifier: A Tutorial, 102.

ens, 2015] Jacobs, B., Vogels, F., & Piessens, F. (2015). Featherweight VeriFast. *Logical Methods in Computer Science*, *11*(3). https://doi.org/10.2168/LMCS-11(3:19)2015

Miné, n.d.] Jeannet, B., & Miné, A. (n.d.). APRON numerical abstract domain library. Retrieved February 1, 2019, from http://apron.cri.ensmp.fr/library/

ung, n.d.] Jung, R. (n.d.). Iris Project. Retrieved February 1, 2019, from https://iris-project.org/

yer, 2017] Jung, R., Jourdan, J.-H., Krebbers, R., & Dreyer, D. (2017). RustBelt: securing the foundations of the rust programming language. *Proceedings of the ACM on Programming Languages*, *2*, 1–34. https://doi.org/10.1145/3158154

al., 2018] Jung, R., Krebbers, R., Jourdan, J.-H., Bizjak, A., Birkedal, L., & Dreyer, D. (2018). Iris from the ground up: A modular foundation for higher-order concurrent separation logic. *Journal of Functional Programming*, *28*. https://doi.org/10.1017/S0956796818000151

iani, n.d.] Kaiser, J.-O., & Ziliani, B. (n.d.). A "destruct" Tactic for Mtac2 - POPL 2018. Retrieved February 1, 2019, from https://popl18.sigplan.org/event/coqpl-2018-a-destruct-tactic-for-mtac2

g et al., 2018] Kang, J., Kim, Y., Song, Y., Lee, J., Park, S., Shin, M. D., … Yi, K. (2018). Crellvm: Verified Credible Compilation for LLVM. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation* (pp. 631–645). New York, NY, USA: ACM. https://doi.org/10.1145/3192366.3192377

ohland, 2015] Kästner, D., & Pohland, J. (2015). Program Analysis on Evolving Software. In M. Roy (Ed.), *CARS 2015 - Critical Automotive applications: Robustness & Safety*. Paris, France. Retrieved from https://hal.archives-ouvertes.fr/hal-01192985

er et al., n.d.] Kästner, D., Wilhelm, S., Nenova, S., Miné, A., Rival, X., Mauborgne, L., … Cousot, R. (n.d.). Astree: Proving the Absence of Runtime Errors, 9. Retrieved from https://www.di.ens.fr/~rival/papers/erts10.pdf

n et al., 2017] Klein Gerwin, Andronick June, Keller Gabriele, Matichuk Daniel, Murray Toby, & O'Connor Liam. (2017). Provably trustworthy systems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *375*(2104), 20150404. https://doi.org/10.1098/rsta.2015.0404

Leino, 2016] Koenig, J., & Leino, R. (2016). Programming Language Features for Refinement. Retrieved from https://www.microsoft.com/en-us/research/publication/programming-language-features-refinement/

pitters, 2011] Krebbers, R., & Spitters, B. (2011). Type classes for efficient exact real arithmetic in Coq. *arXiv:1106.3448 [cs, Math]*. https://doi.org/10.2168/LMCS-9(1:01)2013

mbika, 2018] Krishnan, R., & Lalithambika, V. R. (2018). Modelling and validating 1553B protocol using the SPIN model checker. In *2018 10th International Conference on Communication Systems & Networks (COMSNETS)* (pp. 472–475). Bengaluru: IEEE. https://doi.org/10.1109/COMSNETS.2018.8328247

Kubota, 2016] Kubota, K. (2016). Foundations of Mathematics. https://doi.org/10.4444/100.111

Kubota, n.d.] Kubota, K. (n.d.). Foundations of Mathematics – Owl of Minerva Press. Retrieved February 1, 2019, from http://owlofminerva.net/foundations-of-mathematics/

Rebêlo, 2012] Lahiri, S. K., Hawblitzel, C., Kawaguchi, M., & Rebêlo, H. (2012). SYMDIFF: A Language-Agnostic Semantic Diff Tool for Imperative Programs. In P. Madhusudan & S. A. Seshia (Eds.), *Computer Aided Verification* (pp. 712–717). Springer Berlin Heidelberg.

blitzel, 2015] Lahiri, S. K., Sinha, R., & Hawblitzel, C. (2015). Automatic Rootcausing for Program Equivalence Failures in Binaries. In D. Kroening & C. S. Păsăreanu (Eds.), *Computer Aided Verification* (pp. 362–379). Springer International Publishing.

oort, n.d.] Lamport, L. (n.d.). Specifying Systems. Retrieved February 1, 2019, from https://lamport.azurewebsites.net/tla/book.html

son, 2001] Lampson, B. (2001). The ABCD's of Paxos. In *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing* (p. 13 –). New York, NY, USA: ACM. https://doi.org/10.1145/383962.383969

ard, 1969] Lancaster, E. R., & Blanchard, R. C. (1969). A unified form of lambert's theorem. *NASA Technical Note*, *{TN} D-5368*, 18.

cio, 2015] Leino, K. R. M., & Lucio, P. (2015). An Assertional Proof of the Stability and Correctness of Natural Mergesort. *ACM Trans. Comput. Logic*, *17*(1), 6:1–6:22. https://doi.org/10.1145/2814571

del, 2016] Leino, K. R. M., & Pit-Claudel, C. (2016). Trigger Selection Strategies to Stabilize Program Verifiers. In S. Chaudhuri & A. Farzan (Eds.), *Computer Aided Verification* (Vol. 9779, pp. 361–381). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-41528-4_20

no, 2016a] Leino, R. (2016a). Compiling Hilbert's epsilon Operator. *LPAR-20. 20th International Conferences on Logic for Programming, Artificial Intelligence and Reasoning*, *35*. Retrieved from https://www.microsoft.com/en-us/research/publication/compiling-hilberts-%cf%b5-operator/

no, 2016b] Leino, R. (2016b). Well-Founded Functions and Extreme Predicates in Dafny: A Tutorial, *40*. Retrieved from https://www.microsoft.com/en-us/research/publication/well-founded-functions-extreme-predicates-dafny-tutorial/

kal, 2013] Leino, R., & Moskal, M. (2013). Co-Induction Simply: Automatic Co-Inductive Proofs in a Program Verifier. Retrieved from https://www.microsoft.com/en-us/research/publication/co-induction-simply-automatic-co-inductive-proofs-in-a-program-verifier/

ans, 2016] Leino, R., Müller, P., & Smans, J. (2016). Verification of Concurrent Programs with Chalice. Retrieved from https://www.microsoft.com/en-us/research/publication/verification-concurrent-programs-chalice/

ova, 2016] Leino, R., & Polikarpova, N. (2016). Verified Calculations. Retrieved from https://www.microsoft.com/en-us/research/publication/verified-calculations/

olz, 2016] Leino, R., & Wüstholz, V. (2016). Fine-grained Caching of Verification Results, *9206*. Retrieved from https://www.microsoft.com/en-us/research/publication/fine-grained-caching-verification-results/

ssenov, 2016] Leino, R., & Yessenov, K. (2016). Stepwise Refinement of Heap-Manipulating Code in Chalice. Retrieved from https://www.microsoft.com/en-us/research/publication/stepwise-refinement-heap-manipulating-code-chalice/

[Leroy, n.d.] Leroy, X. (n.d.). OCaml Home Page. Retrieved February 1, 2019, from https://ocaml.org/

etouzey, n.d.] Letouzey, P. (n.d.). Certified functional programming : Program extraction within Coq proof assistant.ResearchGate. Retrieved February 1, 2019, from https://www.researchgate.net/publication/280790704_Certified_functional_programming_Program_extraction_within_Coq_proof_assistant

[Luo, n.d.] Luo, Z. (n.d.). An Extended Calculus of Constructions. Retrieved February 1, 2019, from http://www.lfcs.inf.ed.ac.uk/reports/90/ECS-LFCS-90-118/

alecha, 2018] Malecha, G. (2018). *Reflection library for Coq. Contribute to gmalecha/template-coq development by creating an account on GitHub.* Retrieved from https://github.com/gmalecha/template-coq

quiond, 2016] Martin-Dorel, É., & Melquiond, G. (2016). Proving Tight Bounds on Univariate Expressions with Elementary Functions in Coq. *J Autom Reasoning*, *57*(3), 187–217. https://doi.org/10.1007/s10817-015-9350-4

edilnik, 2013] Martin, K., Hoffman, B., & Cedilnik, A. (2013). *Mastering CMake: a cross-platform build system ; covers installing and running CMake ; details converting existing build processes to CMake ; create powerful cross-platform build scripts* (6. ed). Clifton Park, NY: Kitware.

quiond, n.d.] Melquiond, G. (n.d.). Why3. Retrieved February 1, 2019, from http://why3.lri.fr/

é et al., 2016] Miné, A., Mauborgne, L., Rival, X., Feret, J., Cousot, P., Kästner, D., … Ferdinand, C. (2016). Taking Static Analysis to the Next Level: Proving the Absence of Run-Time Errors and Data Races with Astrée. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*. Toulouse, France. Retrieved from https://hal.archives-ouvertes.fr/hal-01271552

Hickey, n.d.] Minsky, Y., Madhavapeddy, A., & Hickey, J. (n.d.). Real World OCaml. Retrieved February 1, 2019, from http://dev.realworldocaml.org/

lokhov, 2017] Mokhov, A. (2017). Algebraic Graphs with Class (Functional Pearl). In *Proceedings of the 10th ACM SIGPLAN International Symposium on Haskell* (pp. 2–13). New York, NY, USA: ACM. https://doi.org/10.1145/3122955.3122956

aux, 2005] Monniaux, D. (2005). The parallel implementation of the Astr\'{e}e static analyzer. *arXiv:cs/0701191*, *3780*, 86–96. https://doi.org/10.1007/11575467_7

kos, 2016] Murawski, A. S., & Tzevelekos, N. (2016). An Invitation to Game Semantics. *ACM SIGLOG News*, *3*(2), 56–67. https://doi.org/10.1145/2948896.2948902

l., 2017a] Nelson, L., Sigurbjarnarson, H., Zhang, K., Johnson, D., Bornholt, J., Torlak, E., & Wang, X. (2017a). Hyperkernel: Push-Button Verification of an OS Kernel. In *Proceedings of the 26th Symposium on Operating Systems Principles* (pp. 252–269). New York, NY, USA: ACM. https://doi.org/10.1145/3132747.3132748

l., 2017b] Nelson, L., Sigurbjarnarson, H., Zhang, K., Johnson, D., Bornholt, J., Torlak, E., & Wang, X. (2017b). Hyperkernel: Push-Button Verification of an OS Kernel - Slides. In *Proceedings of the 26th Symposium on Operating Systems Principles - SOSP '17* (pp. 252–269). Shanghai, China: ACM Press. https://doi.org/10.1145/3132747.3132748

arn, 2015] O'Hearn, P. (2015). From Categorical Logic to Facebook Engineering. In *Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)* (pp. 17–20). Washington, DC, USA: IEEE Computer Society. https://doi.org/10.1109/LICS.2015.11

arn, 2019] O'Hearn, P. (2019). Separation logic. *Communications of the ACM*, *62*(2), 86–95. https://doi.org/10.1145/3211968

ang, 2001] O'Hearn, P., Reynolds, J., & Yang, H. (2001). Local Reasoning about Programs that Alter Data Structures. In L. Fribourg (Ed.), *Computer Science Logic* (pp. 1–19). Springer Berlin Heidelberg.

arn, 2018] O'Hearn, P. W. (2018). Continuous Reasoning: Scaling the impact of formal methods. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science - LICS '18* (pp. 13–25). Oxford, United Kingdom: ACM Press. https://doi.org/10.1145/3209108.3209109

earn, n.d.] O'Hearn, P. W. (n.d.). Peter W O'hearn - acm profile. Retrieved from https://dl.acm.org/author_page.cfm?id=81332519314&coll=DL&dl=ACM&trk=0

akin, n.d.] Pakin, S. (n.d.). The Comprehensive LaTeX Symbol List, 358.

kov, 2000] Parigot, M., & Voronkov, A. (2000). *Logic for Programming and Automated Reasoning: 7th International Conference, LPAR 2000 Reunion Island, France, November 6-10, 2000 Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg.

ners, n.d.] Parkinson, M. J., & Summers, A. J. (n.d.). The Relationship between Separation Logic and Implicit Dynamic Frames. *LNCS*, *6602*, 439–458.

hmed, n.d.-a] Patterson, D., & Ahmed, A. (n.d.-a). On Compositional Compiler Correctness and Fully Abstract Compilation, 3. Retrieved from https://popl18.sigplan.org/event/prisc-2018-on-compositional-compiler-correctness-and-fully-abstract-compilation

hmed, n.d.-b] Patterson, D., & Ahmed, A. (n.d.-b). On Compositional Compiler Correctness and Fully Abstract Compilation - POPL 2018. Retrieved February 1, 2019, from https://popl18.sigplan.org/event/prisc-2018-on-compositional-compiler-correctness-and-fully-abstract-compilation

aulson, 2000] Paulson, L. C. (2000). The Foundation of a Generic Theorem Prover. *arXiv:cs/9301105*. Retrieved from arXiv:cs/9301105

rrisett, 2015] Petcher, A., & Morrisett, G. (2015). The Foundational Cryptography Framework. In R. Focardi & A. Myers (Eds.), *Principles of Security and Trust* (pp. 53–72). Springer Berlin Heidelberg. Retrieved from http://www.cs.cornell.edu/~jgm/papers/FCF.pdf

lliand, 2015] Petiot, G., Kosmatov, N., Botella, B., Giorgetti, A., & Julliand, J. (2015). Your Proof Fails? Testing Helps to Find the Reason. *arXiv:1508.01691 [cs]*. Retrieved from arXiv:1508.01691

lliand, 2018] Petiot, G., Kosmatov, N., Botella, B., Giorgetti, A., & Julliand, J. (2018). How testing helps to diagnose proof failures. *Form Asp Comp*, *30*(6), 629–657. https://doi.org/10.1007/s00165-018-0456-4

Claudel, n.d.] Pit-Claudel, C. (n.d.). Clément Pit-Claudel. Retrieved January 31, 2019, from http://pit-claudel.fr/clement/

hlipala, n.d.] Pit-Claudel, C., Wang, P., Delaware, B., Gross, J., & Chlipala, A. (n.d.). Extensible Extraction of Efficient Imperative Programs with Foreign Functions, Manually Managed Memory, and Proofs, 14. Retrieved from http://pit-claudel.fr/clement/papers/fiat-to-facade.pdf

latzer, 2008] Platzer, A. (2008). Differential Dynamic Logic for Hybrid Systems. *Journal of Automated Reasoning*, *41*(2), 143–189. https://doi.org/10.1007/s10817-008-9103-8

latzer, 2015] Platzer, A. (2015). Differential Game Logic. *ACM Trans. Comput. Logic*, *17*(1), 1:1–1:51. https://doi.org/10.1145/2817824

latzer, 2017] Platzer, A. (2017). A Complete Uniform Substitution Calculus for Differential Dynamic Logic. *Journal of Automated Reasoning*, *59*(2), 219–265. https://doi.org/10.1007/s10817-016-9385-1

atzer, 2018a] Platzer, A. (2018a). Differential Equations & Differential Invariants. In A. Platzer, *Logical Foundations of Cyber-Physical Systems* (pp. 287–322). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-63588-0_10

zer, 2018] Platzer, A. (2018). *Logical Foundations of Cyber-Physical Systems*. Springer International Publishing. Retrieved from https://www.springer.com/gp/book/9783319635873

er, 2018b] Platzer, A. (2018b). *Logical Foundations of Cyber-Physical Systems - Slides*. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-63588-0

tzer, n.d.] Platzer, A. (n.d.). KeYmaera X: Documentation. Retrieved January 31, 2019, from http://www.ls.cs.cmu.edu/KeYmaeraX/documentation.html

gey, 2019] Polikarpova, N., & Sergey, I. (2019). Structuring the Synthesis of Heap-manipulating Programs. *Proc. ACM Program. Lang.*, *3*, 72:1–72:30. https://doi.org/10.1145/3290385

nas, n.d.] Pottier, F., & REgis-Gianas, Y. (n.d.). Menhir Reference Manual (version 20181113). Retrieved February 1, 2019, from http://gallium.inria.fr/~fpottier/menhir/manual.html

al., 2017] Protzenko, J., Zinzindohoué, J.-K., Rastogi, A., Ramananandro, T., Wang, P., Zanella-Béguelin, S., … Swamy, N. (2017). Verified Low-level Programming Embedded in F*. *Proc. ACM Program. Lang.*, *1*, 17:1–17:29. https://doi.org/10.1145/3110261

eshi, n.d.] Qureshi, Z. H. (n.d.). Formal Modelling and Analysis of Mission-Critical Software in Military Avionics Systems. *11th Australian Workshop on Safety Related Programmable Systems (SCS'06)*, 11. Retrieved from http://crpit.com/confpapers/CRPITV69Qureshi.pdf

ias, 2006] Ramsey, N., & Dias, J. (2006). An Applicative Control-Flow Graph Based on Huet's Zipper. *Electronic Notes in Theoretical Computer Science*, *148*(2), 105–126. https://doi.org/10.1016/j.entcs.2005.11.042

an, 2017] Sherman, B. (2017). *Making Discrete Decisions Based on Continuous Values* (Master of Science). MIT, Cambridge, MA. Retrieved from http://adam.chlipala.net/theses/sherman_sm.pdf

ght, 2009] Shrobe, H., DeHon, A., & Knight, T. (2009). Trust-Management, Intrusion-Tolerance, Accountability, and Reconstitution Architecture (TIARA), 133. Retrieved from https://apps.dtic.mil/dtic/tr/fulltext/u2/a511350.pdf

an, 2013] Shulman, M. (2013, March 12). The HoTT Book.Homotopy Type Theory. Retrieved February 1, 2019, from https://homotopytypetheory.org/book/

eau, 2010] Sozeau, M. (2010). Equations: A Dependent Pattern-Matching Compiler. In M. Kaufmann & L. C. Paulson (Eds.), *Interactive Theorem Proving* (pp. 419–434). Springer Berlin Heidelberg.

eau, 2019] Sozeau, M. (2019). *MetaCoq - Metaprogramming in Coq (Was template-coq)*. MetaCoq. Retrieved from https://github.com/MetaCoq/metacoq

ozeau, n.d.-a] Sozeau, M. (n.d.-a). Typed Template Coq - POPL 2018. Retrieved February 1, 2019, from https://popl18.sigplan.org/event/coqpl-2018-typed-template-coq

ozeau, n.d.-b] Sozeau, M. (n.d.-b). Typed Template Coq - Slides, 11.

Weirich, 2018] Spector-Zabusky, A., Breitner, J., Rizkallah, C., & Weirich, S. (2018). Total Haskell is Reasonable Coq. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs* (pp. 14–27). New York, NY, USA: ACM. https://doi.org/10.1145/3167092

Appel, 2012] Stewart, G., Beringer, L., & Appel, A. W. (2012). Verified Heap Theorem Prover by Paramodulation. In *Proceedings of the 17th ACM SIGPLAN International Conference on Functional Programming* (pp. 3–14). New York, NY, USA: ACM. https://doi.org/10.1145/2364527.2364531

[Swamy, n.d.] Swamy, N. (n.d.). Project Everest - Verified Secure Implementations of the HTTPS Ecosystem.Microsoft Research. Retrieved February 1, 2019, from https://www.microsoft.com/en-us/research/project/project-everest-verified-secure-implementations-https-ecosystem/

ivshits, 2013] Swamy, N., Chen, J., & Livshits, B. (2013). Verifying Higher-order Programs with the Dijkstra Monad. Retrieved from https://www.microsoft.com/en-us/research/publication/verifying-higher-order-programs-with-the-dijkstra-monad/

y et al., 2016] Swamy, N., Hriţcu, C., Keller, C., Rastogi, A., Delignat-Lavaud, A., Forest, S., … Zanella-Béguelin, S. (2016). Dependent Types and Multi-monadic Effects in F*. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (pp. 256–270). New York, NY, USA: ACM. https://doi.org/10.1145/2837614.2837655

Syme, 2019a] Syme, D. (2019a). *Fsharp design: RFCs and docs related to the F# language design process,*. F# Software Foundation Repositories. Retrieved from https://github.com/fsharp/fslang-design

Syme, 2019b] Syme, D. (2019b). *The Fsharp Compiler, Core Library & Tools (F# Software Foundation Repository): fsharp/fsharp*. F# Software Foundation Repositories. Retrieved from https://github.com/fsharp/fsharp

nbuken, 2015] Van Renesse, R., & Altinbuken, D. (2015). Paxos Made Moderately Complex. *ACM Comput. Surv.*, *47*(3), 42:1–42:36. https://doi.org/10.1145/2673577

evodsky, n.d.] Voevodsky, V. (n.d.). Homotopy Type Theory: Univalent Foundations of Mathematics, 490.

Wenzel, 2018] Wenzel, M. (2018). The Isabelle/Isar Reference Manual. Retrieved from https://core.ac.uk/display/22830292

ick, 2017] White Neil, Matthews Stuart, & Chapman Roderick. (2017). Formal verification: will the seedling ever flower? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *375*(2104), 20150402. https://doi.org/10.1098/rsta.2015.0402

ijk, 2008] Wiedijk, F. (2008). Formal Proof—Getting Started, *55*(11), 7.

dia, 2017] Wikipedia. (2017). Category:Formal methods people. In *Wikipedia*. Retrieved from https://en.wikipedia.org/w/index.php?title=Category:Formal_methods_people&oldid=812800009

zel, 2011] Yang, J., & Hawblitzel, C. (2011). Safe to the last instruction: automated verification of a type-safe operating system. *Communications of the ACM*, *54*(12), 123. https://doi.org/10.1145/2043174.2043197

al., 2018] Zhang, T., Wiegley, J., Giannakopoulos, T., Eakman, G., Pit-Claudel, C., Lee, I., & Sokolsky, O. (2018). Correct-by-Construction Implementation of Runtime Monitors Using Stepwise Refinement. In X. Feng, M. Müller-Olm, & Z. Yang (Eds.), *Dependable Software Engineering. Theories, Tools, and Applications* (Vol. 10998, pp. 31–49). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-99933-3_3

[n.d.-a] (n.d.-a). ACM Classification Codes. Retrieved February 1, 2019, from https://cran.r-project.org/web/classifications/ACM.html

[n.d.-b] (n.d.-b). Coq for PL conference series - CoqPL 2019. Retrieved January 31, 2019, from https://popl18.sigplan.org/series/CoqPL

[n.d.-c] (n.d.-c). CoqPL 2018 The Fourth International Workshop on Coq for Programming Languages - POPL 2018. Retrieved January 31, 2019, from https://popl18.sigplan.org/track/CoqPL-2018

[n.d.-d] (n.d.-d). CoqPL 2019 The Fifth International Workshop on Coq for Programming Languages - POPL 2019. Retrieved January 31, 2019, from https://popl19.sigplan.org/track/CoqPL-2019#program

[n.d.-e] (n.d.-e). Frama-C. Retrieved February 1, 2019, from https://frama-c.com/

[n.d.-f] (n.d.-f). LaTeX - Wikibooks, open books for an open world. Retrieved February 1, 2019, from https://en.wikibooks.org/wiki/LaTeX

[n.d.-g] (n.d.-g). MSC2010 database. Retrieved February 1, 2019, from https://mathscinet.ams.org/msc/msc2010.html

[n.d.-h] (n.d.-h). Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. Retrieved February 1, 2019, from https://royalsocietypublishing.org/journal/rsta

[n.d.-i] (n.d.-i). POPL conference series - POPL 2020. Retrieved January 31, 2019, from https://popl18.sigplan.org/series/POPL

[n.d.-j] (n.d.-j). Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences. Retrieved February 1, 2019, from https://royalsocietypublishing.org/journal/rspa