Patrice Godefroid

December 2021

Professional address : Microsoft Research
One Microsoft Way
Redmond, WA 98052, U.S.A.
Email: pg@microsoft.com
http://research.microsoft.com/users/pg/

Employment

October 2006 – present: Partner Researcher, Microsoft Research
(Principal Researcher, promoted to "Microsoft Partner" in September 2012).

December 1994 – August 2006: Distinguished Member of Technical Staff, Bell Laboratories
(Member of Technical Staff, promoted to "Distinguished MTS" in June 2001).

September 1989 – November 1994: Researcher, Computer Science Department (group of Professor Pierre Wolper), Université de Liège, Belgium.
Work on the ESPRIT project SPEC (3096) *Formal Methods and Tools for the Development of Distributed and Real-Time Systems*, on the SPPS Belgian Incentive Program *Information Technology*, and on the ESPRIT project REACT (6021) *Building Correct Reactive Systems*.

July 1992 – August 1992: Visiting researcher, Computing Science Research Center, AT&T Bell Laboratories, Murray Hill, NJ.

Education

September 1989 – November 1994: Ph.D. in Computer Science, Université de Liège, Belgium.
Advisor: Professor Pierre Wolper.

September 1984 – June 1989: Ingénieur Civil Electricien (Informatique) (five-years university degree in Electrical Engineering, Computer Science elective), Université de Liège, Belgium.

Honors

- h-index = 60, 20,000+ citations
(see https://scholar.google.com/citations?user=1bFun-AAAAAJ&hl=en&oi=ao)
- CAV Award 2014 for my work on partial-order reduction (with Doron Peled, Antti Valmari and Pierre Wolper).
- Promoted to Microsoft Partner, September 2012.
- LICS 2011 Test-Of-Time Award for my LICS'1991 paper "A Partial Approach to Model Checking" (with Pierre Wolper).
- HVC 2009 award for "the most promising contribution to fields of software and hardware verification and test in the last five years" for my PLDI'2005 paper "DART: Directed Automated Random Testing" (with Nils Klarlund and Koushik Sen).
- 2008 Médaille d'or du mérite scientifique Gustave Trasenster de l'AILg (University of Liège Alumni Award).
- Promoted to Distinguished Member of Technical Staff, Bell Laboratories, June 2001 (for my work on VeriSoft).
- A revised version of my PhD thesis "Partial-Order Methods for the Verification of Concurrent Systems –*An Approach to the State-Explosion Problem*" is published by Springer-Verlag, as volume 1032 of Lecture Notes in Computer Science, January 1996. (ISBN 3-540-60761-7)
- My undergraduate graduating dissertation "Les modèles ordre-partiel du parallélisme (Partial-Order Models for Concurrency)" received a 1990 IBM Belgium Award for "best graduating dissertation contributing to computer science in Belgium".

Professional Activities

- Referee for international conferences (AAAI, ASE, ATVA, CAV, CONCUR, EMSOFT, FM, FME, FMICS, FORTE, FSE, HVC, ICALP, ICSE, ISSTA, LICS, PASTE, PLDI, POPL, PSTV, RV, SAS, SPIN, TACAS, VMCAI, etc.).
- Referee for professional journals (Journal of the ACM, FML, FMSD, Information and Computation, Information Processing Letters, STTT, TCS, TOCL, TOPLAS, TSE, etc.).
- Referee and panelist for research-funding agencies (NSF, NSERC Canada, INRIA France, Science Foundation Ireland, NWO Netherlands, Israel Science Foundation, Swiss NSF, Icelandic Research Fund, etc.).
- Member of the program committee of international conferences and workshops (ICSE-SEIP'2022, SAS'2021, SPIN'2021, VSTTE'2020, PLDI'2020, SPIN'2019, CAV'2019, SPIN'2018, VSTTE'2018, RV'2017, SPIN'2017, NFM'2017, RV'2016, PLDI'2016, ICSE'2016 V2025, CAV'2015, PLDI'2015 (ERC), TACAS'2015, HVC'2014, ATVA'2014, ISSTA'2014, TACAS'2014, RV'2013, SPIN'2013, ICALP'2013, PLDI'2012, FM'2012, TAP'2012, TACAS'2012, POPL'2012 (ERC), ATVA'2011, RV'2011, NFM'2011, SPIN'2010, ATVA'2010, RV'2010, TAP'2010, MoChArt'2010, MBT'2010, PLDI'2010 (ERC), VMCAI'2010, HVC'2009, TACAS'2009, ISSTA'2009, ASE'2008, FMICS'2008,

LICS'2008, TACAS'2008, RT'2007, FMICS'2007, SPIN'2007, CAV'2007, ISSTA'2007, VM-CAI'2007, LICS'2006, CAV'2006, VMCAI'2006, SPIN'2005 (PC chair), CAV'2005, TACAS'2005, ISSTA'2004, SPIN'2004, CONCUR'2003, TACAS'2003, ICSE'2003, SPIN'2003, POPL'2002, TACAS'2002, FMICS'2002, CAV'2001, CAV'2000, FMSP'2000, CAV'98, etc.).

- Over 50 lectures given at international conferences and seminars.
- Invited keynote speaker at international conferences and workshops (HCSS'2019, TCE'2015, SEFM'2014, ITC'2014, SAS'2014, SEFM'2014, Highlights'2013, FSTTCS'2012, TAP&TOOLS'2011, CSTVA'2011, ISSTA'2010, DMCD'2010, SPIN'2009, MBT'2009, SBMF'2008, RT'2007, IFM'2005, TIME'2005, PASTE'2004, etc.).
- Invited lecturer at international summer schools (Marktoberdorf'2015, Marktoberdorf'2013, Verified Software Shanghai 2012, SAT/SMT'2012, QMC'2012, Vienna Winter School 2012, LASER'2011, SE & Verification Russia 2011, SAT/SMT'2011, Beijing Model Checking 2010, TECS Week 2010, MOVEP'2006, etc.).
- Principal Investigator with Dennis Dams and Kedar Namjoshi of the $640,000 NSF/NASA HDCP project "Analysis Techniques and Tools for Building Robust Software" (NSF CCR-0341658), September 2003 – September 2007.
- Author or co-author of 30+ US patents issued or pending.

Research Overview

My main research area during the last 32 years has been *software model checking* in a broad sense. Here are some main themes. (I also worked on many other smaller projects on other topics that cannot be summarized here – please see my list of publications for more information on those.)

Partial-Order Reduction. In 1988, I started doing research under the supervision of Professor Pierre Wolper on what is now known as *partial-order reduction*. Partial-order reduction (POR) denotes a family of algorithmic techniques for pruning the state spaces of concurrent reactive programs in such a way that parts of the state space that are pruned away are guaranteed not to contain error states of a specific type, such as deadlocks. POR can considerably speed up verification by model checking, and is nowadays implemented in many model checkers. With my advisor Pierre Wolper and collaborators Gerard Holzmann and Didier Pirottin, we pioneered the development of POR, together with other prominent contributors such as Doron Peled and Antti Valmari. My own work on POR culminated in January 1996 with the publication of my PhD thesis in Springer's *Lecture Notes in Computer Science* series. This work was the co-recipient of the CAV 2014 Award.

Software Model Checking via Systematic Testing: VeriSoft. A year or so after joining Bell Labs in 1994, I started working on software model checking, that is, how to broaden the scope of model checking from abstract software systems (specified in modeling languages) to concrete software systems (described using programming languages). In 1996, I designed and implemented a first version of VeriSoft, *the first model checker for analyzing software written in mainstream programming languages* such as C and C++. The main idea behind of VeriSoft is simple: like a traditional model checker computes the product of finite-state machines described in some modeling language, VeriSoft computes the product of operating-system (Unix-like) processes described in any programming language. Several technical innovations made this possible: the use of a run-time scheduler to systematically drive process executions, a construct to simulate nondeterminism at run-time, and the use of partial-order reduction to make a search in the state space of concurrent OS processes tractable even without storing any intermediate states in memory. Many of these features have now been adopted in other software model checkers (like Java PathFinder, CMC, CHESS, etc.).

From 1996 to 2001, I worked mostly on developing VeriSoft further. With several Bell Labs colleagues, we investigated many extensions, described in several published papers. We also started applying VeriSoft to check the correctness of various software applications inside Lucent Technologies. After several successful applications, VeriSoft was also considered in 1999 for a Lucent-funded start-up. Working with Lucent's venture-capitalist team was an enlightening experience. That same year, I taught a course (mostly on VeriSoft) at Stanford University hosted by David Dill, another valuable experience (teaching a new course takes a lot of time!). Meanwhile, VeriSoft was gaining traction in some development and testing groups where its use contributed to finding several expensive, customer-visible bugs in various Lucent products. Since 1999, VeriSoft has also been publicly available outside Lucent and has been licensed to hundreds of users in industry and academia in more than 25 countries. In 2001, I was promoted to "Distinguished Member of the Technical Staff" of Bell Labs for essentially my work on VeriSoft and its successful applications in Lucent.

Software Model Checking via May/Must Abstractions. Around 2000, another approach to software model checking started to emerge: the static approach. Unlike VeriSoft, static software model checkers parse the source code of the software to be checked, compute a conservative abstraction of the original code, and then perform model checking on this abstraction. One of the pioneering projects in that area is SLAM, a static software model checker for C code, developed at Microsoft Research originally by Tom Ball and Sriram Rajamani. From 2001 to 2004, with Glenn Bruns, Radha Jagadeesan and Michael Huth, we developed a new framework for static software model checking that uses may/must abstractions instead of may-only conservative abstractions. With such abstractions, both proofs and counterexamples (bugs) are now guaranteed to be sound, by construction. We also showed that verification results can sometimes be more precise with *generalized model checking*, which checks whether there exists a concretization of an abstraction satisfying a temporal property. From a theoretical point of view, generalized model checking is an interesting problem since it generalizes both model checking (when the abstraction is complete) and satisfiability (when the abstraction is completely unknown), probably the two most studied problems related to temporal logic and verification. From a practical point of view, our work in this area helps explain the foundation of static software model checking.

Automating Software Testing using Program Analysis: DART and SMART. In 2004, I started working with renewed energy on how to extend the VeriSoft approach (aka software model checking via systematic testing) to deal with data-driven applications (after a first attempt described in a PLDI 1998 paper). With Nils Klarlund and Koushik Sen (both funded by my NSF grant with my Bell Labs colleagues Dennis Dams and Kedar Namjoshi), we implemented a first version of *Directed Automated Random Testing*, or DART for short, a new approach to automate testing that combines three main techniques: (1) *automated* interface extraction from source code, (2) *random* testing at that interface, and (3) dynamic test generation to *direct* executions along alternative program paths. The main strength of DART is that testing can be performed completely automatically on any program that compiles, as there is no need to manually write any test driver or harness code. Also, whenever a symbolic expression cannot be generated for an expression involving some input, the concrete value of that input can be used to simplify this expression, which allows dynamic test generation to drive executions through program statements that purely-static test generation cannot handle.

A DART directed search attempts to sweep through all the feasible execution paths of a program using dynamic test generation: the program under test is first executed on some random or well-formed input, symbolic constraints on inputs are gathered at conditional branches during that run, and then a constraint solver is used to generate variants of the previous inputs in order to steer the next execution of the program towards an alternative program branch. This process is repeated until *all* (in practice, many) feasible program paths of the program are executed, while detecting various types of errors using run-time checking tools, like Purify, for instance. DART can thus be viewed as one way of combining static (interface extraction, symbolic execution) and dynamic (testing, run-time checking) program analysis with model-checking techniques (systematic state-space exploration).

Obviously, systematically executing all feasible program paths does not scale to large, realistic programs. In 2006, I developed a variant of the DART search algorithm that performs dynamic test generation *compositionally*. This new algorithm, dubbed SMART, eliminates path explosion due to interprocedural (interblock) paths: the number of whole execution paths becomes linear in the number of intraprocedural paths, instead of being possibly exponential. Moreover, for programs whose conditional statements can all be driven using symbolic execution, this efficiency gain is obtained without losing any precision. A SMART search is key to make the DART approach (i.e., systematic dynamic test generation) scalable to large programs if the goal is to achieve full path coverage (i.e., verification).

The DART technique, also called *dynamic test generation*, *execution-generated tests*, or *concolic testing*, has revolutionized automatic test generation, with thousands of citations to our work and dozens of academic and industrial tools implementing this approach. This work was the recipient of the HVC 2009 Award.

Whitebox Fuzzing for Security Testing: SAGE. In 2006, I joined Microsoft Research and started working on the "*killer app*" for DART, namely *fuzzing*. Fuzzing, or fuzz testing, is the process of finding security vulnerabilities in input-parsing code by repeatedly testing the parser with modified, or fuzzed, inputs. With Michael Levin and several other Microsoft colleagues including (then-intern) David Molnar, we started developing SAGE, the first *whitebox fuzzer* for security testing. Whitebox fuzzing extends DART from unit testing to security testing of large programs. SAGE performs dynamic symbolic execution at the x86 binary level, and implements several optimizations that are crucial for dealing with huge execution traces with hundreds of millions of machine instructions, in order to scale to large file parsers embedded in applications with millions of lines of code, like Microsoft Excel or PowerPoint. SAGE also pioneered the use of search heuristics based on code coverage for fuzzing purposes.

Since 2008, SAGE has been running in production for over 1,000 machine-years, automatically fuzzing hundreds of applications. This is the largest computational usage ever for any Satisfiability-Modul-Theories (SMT) solver according to the authors of the Z3 SMT solver (also from Microsoft Research), with around 10 billion constraints processed to date. During all this fuzzing, SAGE found many new security vulnerabilities (buffer overflows) in hundreds of Windows parsers and Office applications, including image processors, media players, file decoders, and document parsers. Notably, SAGE found roughly one third of all the bugs discovered by file fuzzing during the development of Microsoft's Windows 7, saving (many) millions of dollars by avoiding expensive security patches for nearly a billion PCs worldwide. In 2012, I was promoted to "Microsoft Partner" essentially for my work on SAGE and its successful applications in Microsoft.

Today, whitebox fuzzing has been adopted in many other security-testing tools, and has inspired numerous variants (such as greybox fuzzing and hybrid fuzzing) and extensions. Our seminal work on whitebox fuzzing (first published in 2008) was credited to introducing the "fuzzing" problem to the program analysis, software engineering, and security academic communities, with thousands of citations to our work.

Fuzzing in the Cloud: Project Springfield. In 2015, with my Microsoft Research colleague David Molnar, I co-founded Project Springfield, the *first commercial cloud fuzzing service*. Customers who subscribe to this cloud service can submit fuzzing jobs targeting their own software. Fuzzing jobs are processed by creating many virtual machines in the cloud and by running different fuzzing tools (including SAGE) and configurations on each of these machines. Fuzzing results (bugs) are continually collected by the service and post-processed for analysis, triage and prioritization, with final results available directly to customers on a secured website.

Project Springfield operated as a "virtual start-up" (or "special project") inside Microsoft Research. I served as its CTO for 2 years. Project Springfield was renamed *Microsoft Security Risk Detection* in 2017. Later, the project gradually re-focused on its core technical contributions, in contrast to its initial business aspirations, and evolved into a cloud fuzzing platform called *OneFuzz*, which became open-source in 2020.

Fuzzing the Cloud: RESTler. In 2017, with my Microsoft Research colleague Marina Polishchuk and intern Vaggelis Atlidakis, we started developing RESTler, the *first stateful REST API fuzzing tool* for automatically testing cloud services through their REST APIs and finding security and reliability bugs in these services. For a given cloud service with an OpenAPI/Swagger specification, RESTler analyzes its entire specification, and then generates and executes tests that exercise the service through its REST API. RESTler intelligently infers producer-consumer dependencies among request types from the API specification. During testing, it checks for specific classes of bugs and dynamically learns how the service behaves from prior service responses. This intelligence allows RESTler to explore deeper service states reachable only through specific request sequences (hence the term "stateful") and to find more bugs.

In 2020, RESTler became open-source, and its usage has been steadily growing since, both inside and outside Microsoft. Inside Microsoft, RESTler has found 100s of new bugs in Microsoft Azure, Office365 and Bing services, including severe critical bugs. At the time of this writing, RESTler is still under active development.

Publications: Refereed Conferences

- X. Ge, B. Niu, R. Brotzman, Y. Chen, H. Han, P. Godefroid, and W. Cui. HyperFuzzer: An Efficient Hybrid Fuzzer for Virtual CPUs. In *Proceedings of CCS'2021 (ACM Conference on Computer and Communications Security)*, November 2021.

- D. Gonzalez, T. Zimmermann, P. Godefroid, and M. Schafer. Anomalicious: Automated Detection of Anomalous and Potentially Malicious Commits on GitHub. In *Proceedings of ICSE'2021 (International Conference on Software Engineering), Software-Engineering-in-Practice Track*, Madrid/virtual, May 2021.

- P. Godefroid, B.-Y. Huang, and M. Polishchuk. Intelligent REST API Data Fuzzing. In *Proceedings of ESEC/FSE'2020 (ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering)*, Sacramento/virtual, November 2020.

- V. Atlidakis, P. Godefroid, and M. Polishchuk. Checking Security Properties of Cloud Service REST APIs. In *Proceedings of ICST'2020 (IEEE International Conference on Software Testing, Verification and Validation)*, Porto/virtual, March 2020.

- W. Lam, P. Godefroid, S. Nath, A. Santhiar, and S. Thummalapenta. Root Causing Flaky Tests in a Large-Scale Industrial Setting. In *Proceedings of ISSTA'2019 (International Symposium on Software Testing and Analysis)*, pages 101–111, Beijing, July 2019.

- V. Atlidakis, P. Godefroid, and M. Polishchuk. RESTler: Stateful REST API Fuzzing. In *Proceedings of ICSE'2019 (41st International Conference on Software Engineering)*, pages 748–758, Montreal, Quebec, Canada, 2019.

- K. Bottinger, P. Godefroid, and R. Singh. Deep Reinforcement Fuzzing. In *Proceedings of DLS'2018 (1st Deep Learning and Security Workshop)*, San Francisco, May 2018.

- P. Godefroid, H. Peleg, and R. Singh. Learn&Fuzz: Machine Learning for Input Fuzzing. In *32nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2017)*, November 2017.

- M. Christakis, P. Emmisberger, P. Godefroid, and P. Muller. A General Framework for Dynamic Stub Injection. In *Proceedings of ICSE'2017 (39th International Conference on Software Engineering*, pages 586–596, Buenos Aires, May 2017.

- P. Godefroid. Between Testing and Verification: Dynamic Software Model Checking. In *Proceedings of Marktoberdorf'2015 (Dependable Software Systems Engineering, NATO Science for Peace and Security Series, IOS Press 2016)*, pages 99–116, Marktoberdorf, August 2015.

- M. Christakis and P. Godefroid. IC-Cut: A Compositional Search Strategy for Dynamic Test Generation. In *Proceedings of SPIN'2015 (22nd International SPIN Symposium on Model Checking of Software)*, volume 9232 of *Lecture Notes in Computer Science*, pages 300–318, Stellenbosch, August 2015. Springer-Verlag.

- N. Lopez, N. Bjorner, P. Godefroid, K. Jayaraman, and G. Varghese. Checking Beliefs in Dynamic Networks. In *Proceedings of NSDI'2015 (12th USENIX Symposium on Networked Systems Design and Implementation)*, Oakland, May 2015.

- M. Christakis and P. Godefroid. Proving Memory Safety of the ANI Windows Image Parser using Compositional Exhaustive Testing. *To appear in the Proceedings of VMCAI'2015 (International Conference on Verification, Model Checking and Abstract Interpretation)*, Lecture Notes in Computer Science, Mumbai, January 2015. Springer-Verlag.

- P. Godefroid. Micro Execution. In *Proceedings of ICSE'2014 (International Conference on Software Engineering)*, pages 539–549, Hyderabad, June 2014. ACM.

- P. Godefroid. May/Must Abstraction-Based Software Model Checking for Sound Verification and Falsification. In *Proceedings of Marktoberdorf'2013 (NATO Advanced Study Institute on Software Systems Safety)*, Marktoberdorf, August 2013.

- E. Bounimova, P. Godefroid, and D. Molnar. Billions and Billions of Constraints: Whitebox Fuzz Testing in Production. In *Proceedings of ICSE'2013 (35th International Conference on Software Engineering)*, pages 122–131, San Francisco, May 2013. ACM.

- P. Godefroid and M. Yannakakis. Analysis of Boolean Programs. In *Proceedings of TACAS'2013 (Tools and Algorithms for the Construction and Analysis of Systems)*, volume 7795 of *Lecture Notes in Computer Science*, pages 214–229, Rome, March 2013. Springer-Verlag.

- P. Godefroid. Test Generation Using Symbolic Execution. In *Proceedings of FSTTCS'2012 (IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science)*, pages 24–33, Hyderabad, December 2012.

- P. Godefroid and S. K. Lahiri. From Program to Logic: An Introduction. In *Proceedings of the LASER'2011 Summer School*, volume 7682 of *Lecture Notes in Computer Science*, pages 31–44, Elba, December 2012. Springer-Verlag.

- P. Godefroid and A. Taly. Automated Synthesis of Symbolic Instruction Encodings from I/O Samples. In *PLDI'2012 (ACM SIGPLAN 2012 Conference on Programming Language Design and Implementation)*, pages 441–452, Beijing, June 2012.

- P. Godefroid, S. K. Lahiri, and C. Rubio-Gonzalez. Statically Validating Must Summaries for Incremental Compositional Dynamic Test Generation. In *Proceedings of SAS'2011 (18th International Static Analysis Symposium)*, volume 6887 of *Lecture Notes in Computer Science*, pages 112–128, Venice, September 2011. Springer-Verlag.

- P. Godefroid and D. Luchaup. Automatic Partial Loop Summarization in Dynamic Test Generation. In *Proceedings of ISSTA'2011 (ACM SIGSOFT International Symposium on Software Testing and Analysis)*, pages 23–33, Toronto, July 2011.

- P. Godefroid. Higher-Order Test Generation. In *PLDI'2011 (ACM SIGPLAN 2011 Conference on Programming Language Design and Implementation)*, pages 258–269, San Jose, June 2011.

- C. Cadar, P. Godefroid, S. Khurshid, C.S. Pasareanu, K. Sen, N.Tillmann, and W. Visser. Symbolic Execution for Software Testing in Practice – Preliminary Assessment. In *ICSE'2011*, Honolulu, May 2011.

- P. Godefroid and J. Kinder. Proving Memory Safety of Floating-Point Computations by Combining Static and Dynamic Program Analysis. In *Proceedings of ISSTA'2010 (ACM SIGSOFT International Symposium on Software Testing and Analysis)*, pages 1–11, Trento, July 2010.

- P. Godefroid, A.V. Nori, S.K. Rajamani, and S.D. Tetali. Compositional May-Must Program Analysis: Unleashing The Power of Alternation. In *Proceedings of POPL'2010 (37th ACM Symposium on Principles of Programming Languages)*, pages 43–55, Madrid, January 2010.
- B. Elkarablieh, P. Godefroid, and M.Y. Levin. Precise Pointer Reasoning for Dynamic Test Generation. In *Proceedings of ISSTA'09 (ACM SIGSOFT International Symposium on Software Testing and Analysis)*, pages 129–139, Chicago, July 2009.
- P. Godefroid and N. Piterman. LTL Generalized Model Checking Revisited. In *Proceedings of VMCAI'2009 (10th Conference on Verification, Model Checking and Abstract Interpretation)*, Lecture Notes in Computer Science, Savannah, January 2009. Springer-Verlag.
- K. Etessami and P. Godefroid. An Abort-Aware Model of Transactional Programming. In *Proceedings of VMCAI'2009 (10th Conference on Verification, Model Checking and Abstract Interpretation)*, Lecture Notes in Computer Science, Savannah, January 2009. Springer-Verlag.
- P. Godefroid, M.Y. Levin, and D. Molnar. Active Property Checking. In *Proceedings of EMSOFT'2008 (8th Annual ACM & IEEE Conference on Embedded Software)*, pages 207–216, Atlanta, October 2008. ACM Press.
- R. Xu, , P. Godefroid, and R. Majumdar. Testing for Buffer Overflows with Length Abstraction. In *Proceedings of ISSTA'08 (ACM SIGSOFT International Symposium on Software Testing and Analysis)*, pages 27–38, Seattle, July 2008.
- P. Godefroid and N. Nagappan. Concurrency at Microsoft - An Exploratory Survey. In *$(EC)^2$ (CAV 2008 Workshop on "Exploiting Concurrency Efficiently and Correctly")*, July 2008.
- P. Godefroid, A. Kiezun, and M. Y. Levin. Grammar-based Whitebox Fuzzing. In *Proceedings of PLDI'2008 (ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation)*, pages 206–215, Tucson, June 2008.
- P. Godefroid, M.Y. Levin, and D. Molnar. Automated Whitebox Fuzz Testing. In *Proceedings of NDSS'2008 (Network and Distributed Systems Security)*, pages 151–166, San Diego, February 2008.
- P. Godefroid. Compositional Dynamic Test Generation. In *Proceedings of POPL'2007 (34th ACM Symposium on Principles of Programming Languages)*, pages 47–54, Nice, January 2007.
- A. Chakrabarti and P. Godefroid. Software Partitioning for Effective Automated Unit Testing. In *Proceedings of EMSOFT'2006 (6th Annual ACM & IEEE Annual Conference on Embedded Software)*, pages 262–271, Seoul, October 2006. ACM Press.
- P. Godefroid and N. Klarlund. Software Model Checking: Searching for Computations in the Abstract or the Concrete (Invited Paper). In *Proceedings of IFM'2005 (Fifth International Conference on Integrated Formal Methods)*, volume 3771 of *Lecture Notes in Computer Science*, pages 20–32, Eindhoven, November 2005. Springer-Verlag.
- P. Godefroid, N. Klarlund, and K. Sen. DART: Directed Automated Random Testing. In *Proceedings of PLDI'2005 (ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation)*, pages 213–223, Chicago, June 2005.

- P. Godefroid. The Soundness of Bugs is What Matters (Position Paper). In *Proceedings of BUGS'2005 (PLDI'2005 Workshop on the Evaluation of Software Defect Detection Tools)*, Chicago, June 2005.
- P. Godefroid and M. Huth. Model Checking Vs. Generalized Model Checking: Semantic Minimizations for Temporal Logics. In *Proceedings of LICS'2005 (20th IEEE Symposium on Logic in Computer Science)*, pages 158–167, Chicago, June 2005.
- C. Flanagan and P. Godefroid. Dynamic Partial-Order Reduction for Model Checking Software. In *Proceedings of POPL'2005 (32nd ACM Symposium on Principles of Programming Languages)*, pages 110–121, Long beach, January 2005.
- L. de Alfaro, P. Godefroid, and R. Jagadeesan. Three-Valued Abstractions of Games: Uncertainty, but with Precision. In *Proceedings of LICS'2004 (19th IEEE Symposium on Logic in Computer Science)*, pages 170–179, Turku, July 2004.
- G. Bruns and P. Godefroid. Model Checking with Multi-Valued Logics. In *Proceedings of ICALP'2004 (31st International Colloquium on Automata, Languages and Programming)*, volume 3142 of *Lecture Notes in Computer Science*, pages 281–293, Turku, July 2004. Springer-Verlag.
- P. Godefroid. Reasoning about Abstract Open Systems with Generalized Module Checking. In *Proceedings of EMSOFT'2003 (3rd Conference on Embedded Software)*, volume 2855 of *Lecture Notes in Computer Science*, pages 223–240, Philadelphia, October 2003. Springer-Verlag.
- P. Godefroid and R. Jagadeesan. On the Expressiveness of 3-Valued Models. In *Proceedings of VMCAI'2003 (4th Conference on Verification, Model Checking and Abstract Interpretation)*, volume 2575 of *Lecture Notes in Computer Science*, pages 206–222, New York, January 2003. Springer-Verlag.
- P. Godefroid and R. Jagadeesan. Automatic Abstraction Using Generalized Model Checking. In *Proceedings of CAV'2002 (14th Conference on Computer Aided Verification)*, volume 2404 of *Lecture Notes in Computer Science*, pages 137–150, Copenhagen, July 2002. Springer-Verlag.
- M. Benedikt, J. Freire, and P. Godefroid. VeriWeb: Automatically Testing Dynamic Web Sites. In *Proceedings of WWW'2002 (11th International World Wide Web Conference)*, Honolulu, May 2002.
- S. Chandra, P. Godefroid, and C. Palm. Software Model Checking in Practice: An Industrial Case Study. In *Proceedings of ICSE'2002 (24th International Conference on Software Engineering)*, pages 431–441, Orlando, May 2002. ACM.
- P. Godefroid and S. Khurshid. Exploring Very Large State Spaces Using Genetic Algorithms. In *Proceedings of TACAS'2002 (8th Conference on Tools and Algorithms for the Construction and Analysis of Systems)*, Grenoble, April 2002.
- P. Godefroid, M. Huth, and R. Jagadeesan. Abstraction-based Model Checking using Modal Transition Systems. In *Proceedings of the 12th International Conference on Concurrency Theory* (CONCUR2001), Aalborg, August 2001.

- A. P. Sistla and P. Godefroid. Symmetry and Reduced Symmetry in Model Checking. In *Proceedings of the 13th Conference on Computer Aided Verification* (CAV2001), Paris, July 2001.
- M. Benedikt, P. Godefroid, and T. Reps. Model Checking of Unrestricted Hierarchical State Machines. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming* (ICALP2001), Crete, Greece, July 2001.
- G. Bruns and P. Godefroid. Temporal Logic Query Checking. In *Proceedings of the 11th IEEE Symposium on Logic in Computer Science* (LICS2001), pages 409–417, Boston, June 2001.
- P. Godefroid, J. Herbsleb, L. Jagadeesan, and D. Li. Ensuring Privacy in Presence Awareness Systems: An Automated Verification Approach. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work* (CSCW2000), Philadelphia, December 2000.
- P. Godefroid, L. Jagadeesan, R. Jagadeesan, and K. Laufer. Automated Systematic Testing for Constraint-Based Interactive Services. In *Proceedings of the 8th International Symposium on the Foundations of Software Engineering* (FSE2000), pages 40–49, San Diego, November 2000.
- G. Bruns and P. Godefroid. Generalized Model Checking: Reasoning about Partial State Spaces. In *Proceedings of the 11th International Conference on Concurrency Theory* (CONCUR2000), volume 1877 of *Lecture Notes in Computer Science*, pages 168–182, University Park, August 2000. Springer-Verlag.
- P. Godefroid. Exploiting Symmetry when Model-Checking Software. In *Proceedings of FORTE /PSTV'99 (Formal Description Techniques and Protocol Specification, Testing and Verification)*, pages 257–275, Beijing, October 1999.
- G. Bruns and P. Godefroid. Model Checking Partial State Spaces with 3-Valued Temporal Logics. In *Proceedings of the 11th Conference on Computer Aided Verification* (CAV99), volume 1633 of *Lecture Notes in Computer Science*, pages 274–287, Trento, July 1999. Springer-Verlag.
- C. Colby, P. Godefroid, and L. J. Jagadeesan. Automatically Closing Open Reactive Programs. In *Proceedings of 1998 ACM SIGPLAN Conference on Programming Language Design and Implementation* (PLDI98), pages 345–357, Montreal, June 1998. ACM Press.
- P. Godefroid, R. S. Hanmer, and L. J. Jagadeesan. Model Checking Without a Model: An Analysis of the Heart-Beat Monitor of a Telephone Switch using VeriSoft. In *Proceedings of ACM SIGSOFT International Symposium on Software Testing and Analysis* (ISSTA98), pages 124–133, Clearwater Beach, March 1998.
- B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The Power of QDDs. In *Proceedings of the Fourth International Static Analysis Symposium* (SAS97), Lecture Notes in Computer Science, Paris, September 1997. Springer-Verlag.
- P. Godefroid. VeriSoft: A Tool for the Automatic Analysis of Concurrent Reactive Software (short paper). In *Proc. 9th Conference on Computer Aided Verification* (CAV97), volume 1254 of *Lecture Notes in Computer Science*, pages 476–479, Haifa, June 1997. Springer-Verlag.

- B. Boigelot and P. Godefroid. Automatic Synthesis of Specifications from the Dynamic Observation of Reactive Programs. In *Proceedings of the Third International Workshop on Tools and Algorithms for the Construction and Analysis of Systems* (TACAS97), volume 1217 of *Lecture Notes in Computer Science*, pages 321–333, Twente, April 1997. Springer-Verlag.
- P. Godefroid. Model Checking for Programming Languages using VeriSoft. In *Proceedings of the 24th ACM Symposium on Principles of Programming Languages* (POPL97), pages 174–186, Paris, January 1997.
- P. Godefroid and D. E. Long. Symbolic Protocol Verification with Queue BDDs. In *Proceedings of the 11th IEEE Symposium on Logic in Computer Science* (LICS96), pages 198–206, New Brunswick, July 1996.
- B. Boigelot and P. Godefroid. Symbolic Verification of Communication Protocols with Infinite State Spaces using QDDs. In *Proc. 8th Conference on Computer Aided Verification* (CAV96), volume 1102 of *Lecture Notes in Computer Science*, pages 1–12, New Brunswick, August 1996. Springer-Verlag.
- P. Godefroid. On the Costs and Benefits of using Partial-Order Methods for the Verification of Concurrent Systems (Invited Paper). In *Proceedings of DIMACS Workshop on Partial-Order Methods in Verification*, Princeton, July 1996. AMS.
- B. Boigelot and P. Godefroid. Model Checking in Practice: An Analysis of the ACCESS.bus Protocol using SPIN. In *Proceedings of Formal Methods Europe'96* (FME96), volume 1051 of *Lecture Notes in Computer Science*, pages 465–478, Oxford, March 1996. Springer-Verlag.
- P. Godefroid, D. Peled, and M. Staskauskas. Using Partial-Order Methods in the Formal Validation of Industrial Concurrent Programs. In *Proceedings of International Symposium on Software Testing and Analysis* (ISSTA96), pages 261–269, San Diego, January 1996.
- P. Godefroid. The ULg Partial-Order Package for SPIN (position paper). In *Proceedings of the First SPIN Workshop*, Montreal, October 1995.
- P. Wolper and P. Godefroid. Partial-Order Methods for Temporal Verification (Invited Paper). In *Proc. CONCUR '93*, volume 715 of *Lecture Notes in Computer Science*, pages 233–246, Hildesheim, August 1993. Springer-Verlag.
- P. Godefroid and D. Pirottin. Refining Dependencies Improves Partial-Order Verification Methods. In *Proc. 5th Conference on Computer Aided Verification* (CAV93), volume 697 of *Lecture Notes in Computer Science*, pages 438–449, Elounda, June 1993. Springer-Verlag.
- P. Godefroid and G. J. Holzmann. On the Verification of Temporal Properties. In *Proc. 13th IFIP WG 6.1 International Symposium on Protocol Specification, Testing, and Verification* (PSTV93), pages 109–124, Liège, May 1993. North-Holland.
- P. Godefroid, G. J. Holzmann, and D. Pirottin. State-Space Caching Revisited. In *Proc. 4th Workshop on Computer Aided Verification* (CAV92), volume 663 of *Lecture Notes in Computer Science*, pages 178–191, Montreal, June 1992. Springer-Verlag.

- G. J. Holzmann, P. Godefroid, and D. Pirottin. Coverage Preserving Reduction Strategies for Reachability Analysis. In *Proc. 12th IFIP WG 6.1 International Symposium on Protocol Specification, Testing, and Verification* (PSTV92), pages 349–363, Lake Buena Vista, Florida, June 1992. North-Holland.
- P. Godefroid and P. Wolper. Using Partial Orders for the Efficient Verification of Deadlock Freedom and Safety Properties. In *Proc. 3rd Workshop on Computer Aided Verification* (CAV91), volume 575 of *Lecture Notes in Computer Science*, pages 332–342, Aalborg, July 1991. Springer-Verlag.
- P. Godefroid and P. Wolper. A Partial Approach to Model Checking. In *Proceedings of the 6th IEEE Symposium on Logic in Computer Science* (LICS91), pages 406–415, Amsterdam, July 1991.
- P. Godefroid and F. Kabanza. An Efficient Reactive Planner for Synthesizing Reactive Plans. In *Proceedings of AAAI-91*, volume 2, pages 640–645, Anaheim, July 1991.
- P. Godefroid. Using Partial Orders to Improve Automatic Verification Methods. In *Proc. 2nd Workshop on Computer Aided Verification* (CAV90), volume 531 of *Lecture Notes in Computer Science*, pages 176–185, Rutgers, June 1990. Extended version in ACM/AMS DIMACS Series, volume 3, pages 321–340, 1991.

Publication: Book

- Patrice Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems – An Approach to the State-Explosion Problem*, volume 1032 of *Lecture Notes in Computer Science*. Springer-Verlag, January 1996.

Publications: Refereed Journals[1] and Book Chapters

- P. Godefroid. Fuzzing: Hack, Art, and Science. *Communications of the ACM*, 63(2):70–76, February 2020.
- G. Candea and P. Godefroid. Automated Software Test Generation: Some Challenges, Solutions, and Recent Advances. In *Special Issue LNCS 10000 (Computing and Software Science: State of the Art and Perspectives)*, pages 505–531. Springer-Verlag, 2018.
- P. Godefroid and K. Sen. Combining Model Checking and Testing. In *Handbook of Model Checking*, pages 613–649. Springer-Verlag, 2018.
- P. Godefroid, M.Y. Levin, and D. Molnar. SAGE: Whitebox Fuzzing for Security Testing. *Communications of the ACM*, 55(3):40–44, March 2012.
- P. Godefroid and N. Piterman. LTL Generalized Model Checking Revisited (Invited Paper). *International Journal on Software Tools for Technology Transfer (STTT)*, 13(6):571–584, 2011.

[1] Please note that I do not usually publish in journals unless prompted by an editor or a co-author.

- K. Etessami and P. Godefroid. An Abort-Aware Model of Transactional Programming (Invited Paper). *International Journal on Software Tools for Technology Transfer (STTT)*, 13(6):537–551, 2011.
- P. Godefroid, P. de Halleux, M. Y. Levin, A. V. Nori, S. K. Rajamani, W. Schulte, and N. Tillmann. Automating Software Testing Using Program Analysis. *IEEE Software*, 25(5):30–37, September/October 2008.
- R. Alur, M. Benedikt, K. Etessami, P. Godefroid, T. Reps, and M. Yannakakis. Analysis of Recursive State Machines. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 27(4), July 2005.
- P. Godefroid. Software Model Checking: The VeriSoft Approach (Invited Paper). *Formal Methods in System Design*, 26(2):77–101, March 2005. Also available as Bell Labs Technical Memorandum ITD-03-44189G, March 2003.
- P. Godefroid and S. Khurshid. Exploring Very Large State Spaces Using Genetic Algorithms (Invited Paper). *International Journal on Software Tools for Technology Transfer (STTT)*, 6(2):117–127, August 2004.
- A. P. Sistla and P. Godefroid. Symmetry and Reduced Symmetry in Model Checking. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 26(4):702–734, July 2004.
- P. Godefroid and D. E. Long. Symbolic Protocol Verification with Queue BDDs (Invited Paper). *Formal Methods and System Design*, 14(3):257–271, May 1999.
- B. Boigelot and P. Godefroid. Symbolic Verification of Communication Protocols with Infinite State Spaces using QDDs (Invited Paper). *Formal Methods and System Design*, 14(3):237–255, May 1999.
- P. Godefroid, R. S. Hanmer, and L. J. Jagadeesan. Systematic Software Testing using VeriSoft: An Analysis of the 4ESS Heart-Beat Monitor (Invited Paper). *Bell Labs Technical Journal*, 3(2):32–46, April-June 1998.
- P. Godefroid, D. Peled, and M. Staskauskas. Using Partial-Order Methods in the Formal Validation of Industrial Concurrent Programs (Invited Paper). *IEEE Transactions on Software Engineering*, 22(7):496–507, July 1996.
- P. Godefroid, G. J. Holzmann, and D. Pirottin. State-Space Caching Revisited (Invited Paper). *Formal Methods in System Design*, 7(3):1–15, November 1995.
- P. Godefroid and P. Wolper. A Partial Approach to Model Checking (Invited Paper). *Information and Computation*, 110(2):305–326, May 1994.
- P. Godefroid and P. Wolper. Using Partial Orders for the Efficient Verification of Deadlock Freedom and Safety Properties (Invited Paper). *Formal Methods in System Design*, 2(2):149–164, April 1993. Kluwer Academic Publishers.

Note

Most of the publications listed above are also available from my web-page

`https://patricegodefroid.github.io/`

where they can also be found grouped by themes.