Research Statement

Benjamin Kiesl

November 2019

The main focus of my research is the theory and practice of automated reasoning, aimed at developing techniques that assist us in solving computationally hard problems such as the verification of security protocols, software, and hardware. Many practically relevant problems arising from these fields are simply too complex for us humans, which is why we often ask computers for help. Although computers outclass us when it comes to low-level logical reasoning, they have their own troubles, especially when higher-level arguments are required to solve a problem. Most of my research has therefore revolved around the following question: *How can we enable computers to come up with more abstract arguments, thus turning them into better problem solvers?* I have approached this question from a theoretical and from a practical angle, contributing to the theory of computational logic as well as to the development of powerful automated-reasoning techniques. The contributions I've made together with my co-authors have so far led to four best paper awards and one best paper nomination, yet there are still many open problems I want to tackle.

An example that illustrates the need for more powerful reasoning is the security protocol WPA2, which is implemented in nearly every wireless router, including the one at your home or work place. In 2016, Belgian researchers discovered subtle attacks on WPA2 that allow adversaries to spy on the traffic of Wi-Fi users [24]. A closer look at WPA2 reveals one thing: that the protocol—with the multitude of possible ways it can be interacted with—is simply too complex to be understood by humans, which explains why the mentioned attacks hadn't been discovered by the protocol designers. On the other hand, without the help of humans all current automated approaches also fail at symbolically analyzing WPA2; I have experienced this painfully myself, as I'm currently working on this problem. The development of stronger reasoning techniques is thus a must if we want computers to analyze complex protocols without our help.

In my previous research, I have contributed to both the theory and the practice of automated reasoning itself. Since my move to the CISPA Helmholtz Center for Information Security, my focus has shifted towards the automated analysis of security protocols, a field that poses many intricate problems. My ambition is to apply for research grants that allow me to turn my research ideas, which I'm going to present in this research statement, into practice. At the moment, my concrete plan is to apply for an FWF "*Einzelantrag*" first and after this to try for an FWF *Start* grant. In the long run, my aim is to apply for an ERC grant. In the following, I first give an overview of my previous work and then discuss my current research as well my ideas for the future, both short-term and long-term.

1 Previous Work

As already mentioned, my previous work has dealt with the theory and practice of automated reasoning. When automated-reasoning engines are used to solve problems from real-world domains, the workflow is usually as follows: A problem statement from a real-world domain is encoded into a logical formula that is in turn passed to a reasoning engine. The reasoning engine then tries to solve this formula and—if it succeeds—returns an answer that either directly answers the original problem or that can be transformed into an answer to the original problem. My work on automated reasoning can be broadly divided into three categories:

1.1 Rewriting Logical Formulas

The formulation of real-world problems in the language of logic often yields gigantic formulas that plague even the most efficient reasoning engines. This is especially true for problems arising from software verification [16] or model checking [1]. To deal with this issue, preprocessing techniques are used that rewrite formulas in order to get rid of misleading information so that the reasoning engines can focus on the important aspects of a problem. Together with my co-authors, I have lifted several successful preprocessing techniques from propositional logic to the more general level of first-order logic [14, 13]. In extensive experiments, we could demonstrate that the use of our techniques can boost the performance of state-of-the-art reasoning tools significantly. Our first-order generalization of the *blocked-clause elimination* technique [7] is now part of the theorem prover VAMPIRE [15], which has won the main track of the annual theorem-prover competition (CASC) each year since 2002. We have also developed novel preprocessing techniques for propositional logic that simulate circuit-simplification techniques on logical formulas [9], which can be beneficial in the context of hardware verification.

1.2 Proof Complexity

When we use automated tools for solving problems from different domains, we usually expect them not only to give us simple yes/no answers but to provide us with independently checkable proofs that justify their answers. Over the years, researchers have suggested various so-called *proof systems*, which define both the ways in which proofs can be represented and the kinds of arguments a reasoning engine can use [20]. To shed light on the potential and the restrictions of different proof systems, the theory of *proof complexity* aims at establishing relationships between different proof systems. In collaboration with my co-authors, I have contributed to this area by answering open questions surrounding proof systems used in practice [10, 11, 12]. We have, for instance, shown that the proof system DRAT [8] (the de-facto standard in state-of-the-art reasoning for propositional logic) can be polynomially simulated by the proof system of extended resolution [23] (a well-known proof system invented already in the sixties), thus answering a question that had remained open since the introduction of DRAT. Moreover, we have also invented strong proof systems ourselves, which has led to the development of a new reasoning paradigm for propositional logic, which I discuss in the following.

1.3 A New SAT Solving Paradigm

The problem of reasoning in propositional logic, commonly called *SAT solving*, is the canonical NP-complete problem and is thus considered intractable from the viewpoint of complexity theory. In practice, however, SAT solvers are surprisingly successful: the dominating SAT solving paradigm, called *conflict-driven clause learning* (CDCL) [17], can often handle gigantic formulas from various domains without much trouble. Still, there exist seemingly simple formulas on which even the strongest CDCL solvers fail spectacularly. A reason for this has been provided by researchers in proof complexity, who proved that many of these formulas admit no short arguments within the proof system used by CDCL solvers. To solve these formulas efficiently, it is thus not enough to just optimize the code of existing CDCL solvers; instead, a fundamentally different solving approach—based on another proof system—is required.

My co-authors and I therefore invented new proof systems that allow for shorter arguments while still being close to practical SAT solving [2, 5]. We were able to demonstrate that our proof systems admit short proofs of popular formulas that only admit exponential-size proofs in the proof system used by conventional CDCL solvers. Even more, we could also demonstrate that the correctness of proofs in our proof systems can be validated efficiently by automated proof-checking tools. We then came up with a SAT solving paradigm, called *satisfaction-driven clause learning* (SDCL), that harnesses the strengths of our proof systems in order to deal with formulas that are beyond usual CDCL solvers [6, 4]. As we could demonstrate, our paradigm is able to come up with surprisingly short arguments when solving certain hard formulas [3]. Moreover, our SDCL paradigm is a generalization of CDCL in the sense that every CDCL solver can be turned into an SDCL solver without drastic modifications of the core algorithm.

2 Current and Future Directions

My current research interests revolve around the development of stronger reasoning techniques and their application to real-world problems, with a particular focus on the analysis and verification of security protocols. In the following, I outline my current work as well as my ides for the future. After this, I conclude by stating my long-term research goals.

2.1 A Verified Model of WPA2

In collaboration with Cas Cremers and one of his students, I'm currently developing a formal model of the Wi-Fi protocol WPA2 mentioned earlier. WPA2 enables clients to establish secret keys with Wi-Fi routers in order to protect subsequent communication. To prevent previous attacks on WPA2, fixes were incorporated into the protocol, but there is still no proof that these fixes meet the desired goals. We want to change this by verifying our formal model of WPA2 with the prover *Tamarin* [18]—a state-of-the-art automated-reasoning tool geared towards security protocols. Our work has so far required us to manually specify a plethora of lemmas that help Tamarin prove our main theorems (the main theorems basically state that the protocol has the desired properties); without these lemmas Tamarin is not able to make any progress. A reason for this is that WPA2 is immensely complex, specifying several state machines that interact in subtle ways with each other. Once our work on WPA2 is finished, we want to use the knowledge we've gained in this project to improve the Tamarin tool such that in the future it will derive many of the manually specified lemmas itself.

2.2 Automatic Generation of Lemmas

Fully automatic approaches to automated reasoning often fail at dealing with hard problems because they are unable to derive certain arguments that are required to solve a given reasoning problem. It is therefore a common approach (commonly called *interactive theorem proving*) to allow users to specify these arguments themselves in the form of *lemmas* and then allow the reasoning tool to make use of these lemmas. In order for this approach to be sound, the reasoning tool is usually not only used to solve its original problem but also to show that the lemmas it used are correct too. By taking a closer look at lemmas that commonly need to be specified by users when verifying security protocols, we can observe that many of these lemmas are of a similar form, for example stating certain invariants. Although the automatic generation of such lemmas is a complicated task in general, for security protocols I already have some concrete ideas on how to do this. By developing such lemma-generation approaches and incorporating them into tools like Tamarin, many protocols whose verification currently requires user interaction will become verifiable by fully automatic approaches.

2.3 Combination of Different Reasoning Approaches

When a particular tool is used for verifying a security protocol, we can sometimes observe that the tool fails to deal with some problem even though the problem should—in theory—be solvable without much trouble if only the right forms of logical reasoning are applied. Different automated-reasoning tools use different kinds of reasoning, which is why I believe that we can make our lives easier by incorporating different approaches in a unified way, similar to what is done in the tool *Sledgehammer* [21] for general interactive theorem proving (not geared towards security protocols). To make this work, the automated translation of (formal) protocol models into different logical representations is required, which might be non-trivial but not impossible. The result would be a tool that combines the strengths of different tools to solve complex problems.

2.4 Stronger Reasoning Techniques for SAT Solvers

I mentioned in Section 1.3 that my co-authors and I have come up with a SAT solving paradigm, called SDCL, that can handle formulas which are—due to theoretical restrictions—too hard for ordinary SAT solvers. While we have achieved promising results with this paradigm, many challenges still lie ahead, especially if we want to apply the SDCL paradigm to hard problems from cryptanalysis—a field where SAT solvers have been applied successfully in the past [19, 22]. One challenge is the fine-tuning of the heuristics used in SDCL solvers. To deal with this issue, I'm currently collaborating with Vijay Ganesh (UC Waterloo) and one of his students, Joe Scott, to find ways—possibly based on machine learning—that allow us to find effective heuristics and thus improve the performance of solvers. Another challenge is the effective incorporation of reasoning over XOR equations, since the XOR operator plays an essential role in the construction of most practical cryptographic primitives such as block ciphers (e.g., AES) or hash functions (e.g., SHA-3). A theoretical problem related to this is to develop approaches that allow us to compactly express forms of XOR reasoning (e.g. Gaussian elimination) in the proof system underlying SDCL solvers. Moreover, we have to find ways to combine the XOR reasoning techniques with the other components of a solver in such a way that they can benefit from reach other.

3 Vision

I want to see a world where automated-reasoning tools are powerful enough to deal with hard real-world problems and especially with the verification of complex security protocols. Thereby, these tools use as little human assistance as possible, relying on techniques—possibly based on machine learning—that allow them to think more like humans when this is necessary, but without sacrificing their own strengths.

References

[1] Armin Biere. Bounded model checking. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 457–481. IOS Press, 2009.

[2] Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Short proofs without new variables. In *Proceedings of the 26th International Conference on Automated Deduction (CADE-26)*, volume 10395 of *LNCS*, pages 130–147. Springer, 2017.

[3] Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Clausal proofs of mutilated chessboards. In *Proc. of the 11th Int. NASA Formal Methods Symposium (NFM 2019)*, volume 11460 of *LNCS*, pages 204–210. Springer, 2019.

[4] Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Encoding redundancy for satisfaction-driven clause learning. In *Proc. of the 25th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2019)*, volume 11427 of *LNCS*, pages 41–58. Springer, 2019.

[5] Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Strong extension-free proof systems. *Journal of Automated Reasoning*, Feb 2019.

[6] Marijn J. H. Heule, Benjamin Kiesl, Martina Seidl, and Armin Biere. PRuning through satisfaction. In *Proc. of the 13th International Haifa Verification Conference (HVC 2017)*, volume 10629 of *LNCS*, pages 179–194. Springer, 2017.

[7] Matti Järvisalo, Armin Biere, and Marijn J. H. Heule. Blocked clause elimination. In *Proc. of the 16th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2010)*, volume 6015 of *LNCS*, pages 129–144. Springer, 2010.

[8] Matti Järvisalo, Marijn J. H. Heule, and Armin Biere. Inprocessing rules. In *Proc. of the 6th Int. Joint Conference on Automated Reasoning (IJCAR 2012)*, volume 7364 of *LNCS*, pages 355–370. Springer, 2012.

[9] Benjamin Kiesl, Marijn J. H. Heule, and Armin Biere. Truth assignments as conditional autarkies. In Yu-Fang Chen, Chih-Hong Cheng, and Javier Esparza, editors, *Proc. of the 17th International Symposium on Automated Technology for Verification and Analysis (ATVA 2019)*, volume 11781 of *LNCS*, pages 48–64. Springer, 2019.

[10] Benjamin Kiesl, Marijn J. H. Heule, and Martina Seidl. A little blocked literal goes a long way. In *Proc. of the 20th International Conference on Theory and Applications of Satisfiability Testing* (*SAT 2017*), volume 10491 of *LNCS*, pages 281–297. Springer, 2017.

[11] Benjamin Kiesl, Adrián Rebola-Pardo, and Marijn J. H. Heule. Extended resolution simulates DRAT. In *Proc. of the 9th International Joint Conference on Automated Reasoning* (*IJCAR 2018*), volume 10900 of *LNCS*, pages 516–531. Springer, 2018.

[12] Benjamin Kiesl and Martina Seidl. QRAT polynomially simulates ∀-Exp+Res. In Mikolás Janota and Inês Lynce, editors, *Proc. of the 22nd Internationl Conference on Theory and Applications of Satisfiability Testing* (*SAT 2019*), volume 11628 of *LNCS*, pages 193–202. Springer, 2019.

[13] Benjamin Kiesl and Martin Suda. A unifying principle for clause elimination in first-order logic. In *Proc. of the 26th Int. Conference on Automated Deduction* (*CADE-26*), volume 10395 of *LNCS*, pages 274–290. Springer, 2017.

[14] Benjamin Kiesl, Martin Suda, Martina Seidl, Hans Tompits, and Armin Biere. Blocked clauses in first-order logic. In *Proc. of the 21st Int. Conference on Logic for Programming, Artificial Intelligence and Reasoning* (*LPAR-21*), volume 46 of *EPiC Series in Computing*, pages 31–48. EasyChair, 2017.

[15] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In *Proc. of the 25th International Conference on Computer Aided Verification,* (*CAV 2013*), volume 8044 of *LNCS*, pages 1–35. Springer, 2013.

[16] Daniel Kroening. Software verification. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 505–532. IOS Press, 2009.

[17] João Marques-Silva, Ines Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 131–153. IOS Press, 2009.

[18] Simon Meier, Benedikt Schmidt, Cas Cremers, and David A. Basin. The TAMARIN prover for the symbolic analysis of security protocols. In Natasha Sharygina and Helmut Veith, editors, *Proc. of the 25th International Conference on Computer Aided Verification* (*CAV 2013*), volume 8044 of *LNCS*, pages 696–701. Springer, 2013.

[19] Ilya Mironov and Lintao Zhang. Applications of SAT solvers to cryptanalysis of hash functions. *IACR Cryptology ePrint Archive*, 2006:254, 2006.

[20] Jakob Nordström. On the interplay between proof complexity and SAT solving. *SIGLOG News*, 2(3):19–44, 2015.

[21] Lawrence C. Paulson and Jasmin Christian Blanchette. Three years of experience with sledgehammer, a practical link between automatic and interactive theorem provers. In Geoff Sutcliffe, Stephan Schulz, and Eugenia Ternovska, editors, *Proc. of the 8th International Workshop on the Implementation of Logics* (*IWIL 2010*), volume 2 of *EPiC Series in Computing*, pages 1–11. EasyChair, 2010.

[22] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In Oliver Kullmann, editor, *Proc. of the 12th International Conference on Theory and Applications of Satisfiability Testing* (*SAT 2009*), volume 5584 of *Lecture Notes in Computer Science*, pages 244–257. Springer, 2009.

[23] G. S. Tseitin. On the complexity of derivation in propositional calculus. *Studies in Mathematics and Mathematical Logic*, 2:115–125, 1968.

[24] Mathy Vanhoef and Frank Piessens. Key reinstallation attacks: Forcing nonce reuse in WPA2. In *Proc. of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (*CCS 2017*), pages 1313–1328. ACM, 2017.