In recent years, automated formal verification of software has progressed from a few research labs into large-scale applications, such as cloud infrastructure and smart contracts. Formal verification techniques based on model checking provide the necessary guarantees by exploring systems' behaviour *exhaustively* and *automatically*. Moreover, they provide *witnesses* (explanations) for the result of their analysis: a faulty behaviour, if there exists one, or a proof of the absence of such behaviour.

However, the general problem that automated software verification is trying to solve is undecidable. Despite this theoretical barrier, it is quite efficient on many instances that arise in practice. We ascribe this (perhaps surprising) success to a combination of factors: the relentless effort of researchers that come up with new verification procedures to tackle classes of problems where existing techniques struggle; amazing progress in the foundational technologies of satisfiability solving, especially in *Satisfiability Modulo Theories* (SMT); and increase of available computational power through parallel and cloud computing. Nevertheless, the growing complexity of real-world systems poses new challenges for formal verification, especially for the scalability of the techniques.

The task of automated software verification has two parts: modelling the task in a formal framework and solving the resulting mathematical problem. While modelling is a non-trivial step in the verification process, it has been addressed widely, and there exist numerous modelling concepts suitable for various systems. Solving, on the other hand, is a bottleneck when it comes to complex modern programs. This thesis focuses on the solving part of the task, where there is a need for new effective solutions. We assume the problems are modelled *symbolically*, with formulas in first-order logic. Specifically, we work in the logical framework of *constrained Horn clauses* (CHC) and research the mathematical problem of *deciding satisfiability* of a CHC system. CHC satisfiability generalizes the common task of verifying *safety properties* in *transition systems*, a widespread model in formal verification. This task is complex and undecidable in general already if the language of the constraints contains linear integer arithmetic. In our work, we argue that this task can be approached by providing solutions at different levels, which we identify as *foundational*, *verification* and *cooperative* layers of the problem. These correspond to decision and interpolation procedures, sequential model-checking algorithms, and multi-agent solving approaches. We further argue that the next (higher) layers build on, and interact with, the previous (lower) layers and that working on the higher layers can significantly benefit from a deep understanding of the layers beneath them. Overall, we advance the field of automated software verification by contributing solutions on all three layers.

On the foundational layer, we contribute a new interpolation algorithm for conflicts in the theory of linear arithmetic. It extends the standard approach based on the Farkas lemma and can compute logically stronger interpolants. Experimental evaluation in a model-checking scenario shows that with our interpolation algorithm, the same model-checking algorithm can successfully solve some problems on which it diverges using the original interpolation algorithm.

On the verification layer, we invent the concept of *transition power abstraction* (TPA) sequence and contribute TPA-based model-checking algorithms that address the known problem of detecting deep counterexamples in transition systems. Moreover, we show that the TPA sequence can be mined for candidates for *transition* invariants. This allows TPA-based algorithms to prove systems safe by means largely orthogonal to existing techniques.

To support the development of verification techniques, we contribute Golem, a new solver for the satisfiability of systems of constrained Horn clauses. The main features of Golem are its tight integration with the underlying interpolating SMT solver and support for multiple back-end solving algorithms. Golem is primarily meant to serve as a research tool for further investigation of SMT-based algorithms for model checking and general Horn solving. It was instrumental in developing our prototype implementation of the TPA-based algorithms. However, it is also

efficient compared to other Horn solvers in the latest edition of CHC-COMP. As such, it can be used as the back end for domain-specific tools that model various verification tasks in the CHC framework. It has already been included as a possible back end for the software verifier Korn.

On the cooperative layer, we contribute an abstract framework that generalizes concepts from induction-based model-checking algorithms. The abstraction aims explicitly at the application in a multi-agent solving scenario where multiple instances of the same solver exchange information and, in this way, cooperate to solve a single problem instance. We instantiate the framework to obtain a parallel version of a successful PD-KIND algorithm and experimentally show that exchanging information can significantly improve performance. Since PD-KIND relies on interpolation as a sub-procedure, we use our novel interpolation algorithm to obtain more diverse behaviour of the agents, and this constitutes a large part of the performance improvement.