Interactive theorem provers like PVS, HOL or Isabelle traditionally support rich specification logics. Proof search and automation for these logics however is difficult, and proving non-trivial theorems usually requires manual guidance by an expert user. Automated theorem provers on the other hand, while often designed for simpler logics, have become increasingly powerful over the past few years. New algorithms, improved heuristics and faster hardware allow interesting theorems to be proved with little or no human interaction, sometimes within seconds.

In the latter case, the satisfying assignment is displayed to the user. The assignment constitutes counterexamples for unsolvable conjectures. zChaff is able to produce concrete counterexamples.

To prove a propositional tautology $\phi$ in the Isabelle/HOL system with the help of zChaff, we proceed in several steps. First $\phi$ is negated, and the negation is converted into an equivalent formula $\neg \phi$ in conjunctive normal form. $\neg \phi$ is then written to a file in DIMACS CNF format, the input format supported by zChaff (and many other SAT solvers). zChaff, when run on this file, returns either "unsatisfiable", or a satisfying assignment for $\phi$. In the latter case, the satisfying assignment is displayed to the user. The assignment constitutes a counterexample to the original conjecture. When zChaff returns "unsatisfiable", however, things are more complicated. The LCF-style [NPW02] approach demands that we verify zChaff’s claim of unsatisfiability within Isabelle/HOL. While this is not as simple as the validation of a satisfying assignment, the increasing complexity of SAT solvers has offered better the question of support for independent verification of their results, and in 2003, zChaff has been enhanced by L. Zhang and S. Malik [ZM03] to generate resolution-style proofs that can be verified by an independent checker. Hence our main task boils down to using Isabelle/HOL as an independent checker for the resolution proof found by zChaff.

Standard Resolution

Isabelle/HOL offers higher-order logic, whereas zChaff only supports formulas of propositional logic in conjunctive normal form. Therefore the (negated) input formula $\phi$ must be preprocessed before it can be passed to zChaff. Note that it is not sufficient to convert $\phi$ into an equivalent formula $\neg \phi$ in CNF. Rather, we have to prove this equivalence inside Isabelle/HOL. The result is not a single formula, but a theorem of the form $\phi \equiv \neg \phi$. Our main workhorse for the construction of this theorem is a generic function $\text{ths}_\phi$.

All necessary preprocessing steps can then be handled with proper instantiations for $\text{decomp}$. zChaff treats clauses as sets of literals, making implicit use of associativity, commutativity and idempotence of disjunction. Therefore some further preprocessing is necessary, aside from conversion to CNF, removal of parentheses, of duplicate literals, and of tautological clauses. Each preprocessing step yields an equivalent theorem that was proved in Isabelle/HOL, and transitivity of $\equiv$ allows us to combine these theorems into a single theorem $\phi \equiv \neg \phi$, where $\phi$ is the final result of our conversion.

Proof Reconstruction

Theorem Proving in Isabelle is based on two simple functions: one that uses resolution to derive new theorems of the form $\phi \equiv \psi$ from existing theorems $\phi \equiv \chi$, $\psi \equiv \chi$, and another function that proves $\phi \equiv \psi$ (where $\psi$ is a single literal) from a theorem $\phi \equiv \chi$. Here $\phi$ must be a clause that contains $\chi$, and for all other literals $\eta$ in $\eta$ a theorem of the form $\phi \equiv \eta$ must be convertible.

Proof reconstruction then proceeds in three steps: First the conflict clause is proved by a call to $\text{prove}$. Clause $\text{prove}$. Every literal in the conflict clause, to show that the literal must be false. Finally resolving the conflict clause with these negated literals yields the theorem $\phi \equiv \neg \phi$. For efficiency reasons, the actual implementation is slightly different from what is shown above. Theorems that were proved once are stored in two arrays (one for clauses, one for literals), and simply looked up, rather than re-proved — should they be needed again.

Hence our implementation is not purely functional.

Conclusions and Future Work

Our results show that the zChaff-based tactic is clearly superior to Isabelle’s built-in tactic for propositional formulas. With the help of zChaff, many formulas that were previously out of the scope of Isabelle’s built-in tactic can now be proved — or refuted — automatically, often within seconds. Isabelle’s applicability as a tool for formal verification, where large propositional problems occur in practice, has thereby improved considerably, even though its performance is not yet sufficient to treat huge SAT problems with thousands of clauses.

The approach presented in this paper has applications beyond propositional reasoning. The decision problem for (fragments of) richer logics can be reduced to SAT.

Consequently, proof reconstruction for propositional logic can serve as a foundation for proof reconstruction for other logics. Based on our work, one only needs a proof-generating implementation of the reduction to integrate the whole SAT-based decision procedure with an LCF-style theorem prover.

Acknowledgments

The author would like to thank Sharad Malik and Zhaohui Fu for their help with zChaff, and Tobias Nipkow for his valuable suggestions.

References


