Clausal Proofs of Mutilated Chessboards*

Marijn J.H. Heule[1], Benjamin Kiesl[2,3], and Armin Biere[4]

[1] Department of Computer Science, The University of Texas at Austin, United States
[2] Institute of Logic and Computation, TU Wien, Austria
[3] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
[4] Institute for Formal Models and Verification, JKU Linz, Austria

Abstract. Mutilated chessboard problems have been called a "tough nut to crack" for automated reasoning. They are, for instance, hard for resolution, resulting in exponential runtime of current SAT solvers. Although there exists a well-known short argument for solving mutilated chessboard problems, this argument is based on an abstraction that is challenging to discover by automated-reasoning techniques. In this paper, we present another short argument that is much easier to compute and that can be expressed within the recent (clausal) PR proof system for propositional logic. We construct short clausal proofs of mutilated chessboard problems using this new argument and validate them using a formally-verified proof checker.

Introduction

The success of automated reasoning presents us with an interesting peculiarity: While modern solving tools can routinely handle gigantic real-world instances, they often fail miserably on supposedly easy problems. Their poor performance is frequently caused by the weakness of their underlying proof systems, which only allow them to derive facts that are logically implied. A recently proposed proof system, called PR [6], overcomes this issue by allowing the derivation of facts that are not necessarily implied but whose addition preserves satisfiability.

A well-known family of problems on which traditional reasoning approaches fail are the *mutilated chessboard problems*. Given a chessboard of size $n \times n$ from which two opposite corner squares have been removed (see Fig. 1), a mutilated chessboard problem asks if the remaining squares can be fully covered with dominos (i.e., with stones that cover exactly two squares). The answer is *no*, based on a simple argument: Assume to the contrary that a mutilated chessboard can be fully covered with dominos. Then, since every domino covers exactly one black square and one white square, the number of covered black squares must equal the number of covered white squares. But the number of black squares on a mutilated chessboard is different from the number of white squares since opposite corner squares (of which two were removed) are of the same color.

* Supported by NSF under grant CCF-1813993, by AFRL Award FA8750-15-2-0096, Austrian Science Fund (FWF) under projects W1255-N23 and S11409-N23 (RiSE) and the LIT Secure and Correct Systems Lab funded by the State of Upper Austria.

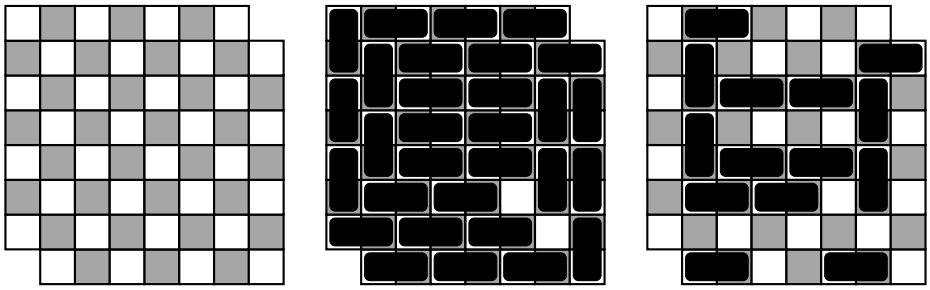

Fig. 1. An empty (left), almost full (middle), and reduced (right) mutilated chessboard.

Automated-reasoning methods on various representations have severe difficulties finding this argument because they do not have colored squares, so they need to come up with this abstraction themselves in order to use a similar argument. John McCarthy has called the mutilated chessboard problems a "tough nut to crack" for automated reasoning [10], and it has been shown that these problems admit only proofs of exponential size within the propositional resolution proof system, which forms the basis of many SAT solvers [1, 3].

In this paper, we show that the recently introduced PR proof system facilitates a completely different but equally short argument for solving mutilated chessboard problems. The new argument rules out possible patterns for the dominos by generalizing—*without loss of generality*—from certain specific patterns that are similar to them. Moreover, the argument also seems to be well suited for automated reasoning since we discovered it when analyzing PR proofs that were found by one of our tools [7]. We argue that the key to automatically solving the mutilated chessboard problems and many other hard problems is not to simulate human thinking but to equip computers with capabilities to find their own short arguments. Moreover, our example demonstrates that automated-reasoning tools cannot only provide us with simple yes/no answers but that they can also help us gain further insights into the nature of a problem.

Representation

A successful approach to solving hard combinatorial problems is to encode them into propositional logic and to solve the resulting propositional formulas with a SAT solver. Mutilated chessboard problems can be naturally encoded in propositional logic: We consider only propositional formulas in conjunctive normal form (CNF). Such formulas are conjunctions of clauses where each clause is a disjunction of literals and each literal is either a variable or the negation of a variable. We use a distinct Boolean variable for each possible placement of a domino on the chessboard. On each square (apart from some border squares), a domino can be placed either horizontally or vertically, having the square either left or top. Intuitively, a variable should be true if and only if a domino is

placed on the corresponding location in the corresponding way (horizontally or vertically). The number of variables is thus roughly twice the number of squares.

Only one constraint needs to be encoded for mutilated chessboard problems: Each square must be covered by exactly one domino. This constraint can be easily expressed in propositional logic, resulting in two clauses for corner squares, four clauses for border squares, and seven clauses for center squares: One clause (consisting only of positive literals) expresses that the square is covered by *at least one* domino whereas the other clauses (each containing exactly two negative literals) express that the square is covered by *at most one* domino.

Mutilated chessboard problems using this encoding were part of the SAT Competition 2018 (in the *main*, *parallel*, and *no-limits* tracks). The problems used in the competition encoded mutilated chessboards of size $n \times n$, with $n \in \{15, 16, 17, 18, 19, 20\}$. None of the solvers were able to solve the two largest instances within the time limit of 5000 seconds. In this paper, we consider significantly larger mutilated chessboards with $n \in \{20, 30, 40, 50\}$.

Notice that the problem encoding does not include any information regarding the colors of squares as in the illustrations. It would be possible to include this information using only a subset of the clauses, allowing at most one domino for white squares and requiring at least one domino for black squares (assuming that the removed corner squares are black). These formulas would still be hard for resolution (using a subset of the clauses cannot yield shorter resolution proofs) but they would become easy for cutting-plane reasoning [8].

Clausal Proofs

Informally, a clausal proof system allows us to show the unsatisfiability of a CNF formula by continuously deriving more and more clauses until we obtain the empty clause. Thereby, the addition of a derived clause to the formula and all previously derived clauses must preserve satisfiability. As the empty clause is trivially unsatisfiable, a clausal proof shows the unsatisfiability of the original formula. Moreover, it must be checkable in polynomial time that each derivation step does preserve satisfiability. This requirement ensures that the correctness of proofs can be efficiently verified. In practice, this is achieved by allowing only the derivation of specific clauses that fulfill some efficiently checkable criterion.

Formally, clausal proof systems are based on the notion of *clause redundancy*. A clause C is *redundant* with respect to a formula F if F and $F \wedge C$ are equisatisfiable (i.e., they are either both satisfiable or both unsatisfiable). Given a formula $F = C_1 \wedge \cdots \wedge C_m$, a *clausal proof* of F is a sequence $(C_{m+1}, \omega_{m+1}), \ldots, (C_n, \omega_n)$ of pairs where each C_i is a clause, each ω_i (called the *witness*) is a string, and C_n is the empty clause [6]. Such a sequence gives rise to formulas $F_m, F_{m+1}, \ldots, F_n$, where $F_i = C_1 \wedge \cdots \wedge C_i$. A clausal proof is *correct* if every clause C_i $(i > m)$ is redundant with respect to F_{i-1}, and if this redundancy can be checked in polynomial time (with respect to the size of the proof) using the witness ω_i.

An example for a clausal proof system is the resolution proof system, which only allows the derivation of resolvents (with no/empty witnesses). Moreover,

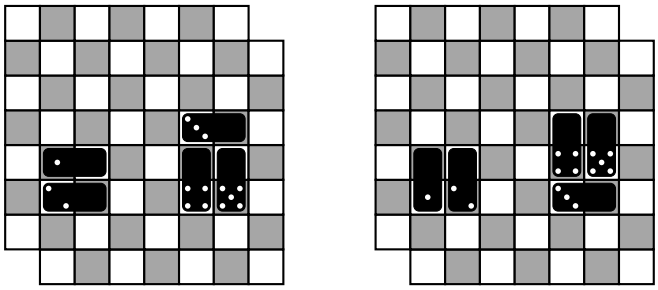

Fig. 2. Two equivalent placements of five dominos on a mutilated chessboard.

the recently introduced proof system PR [6] is a clausal proof system that allows to derive a clause C_i if that clause is *propagation redundant* with respect to F_{i-1}. For the details of propagation redundancy, we refer to the original paper [6]. Here, we just note that (1) propagation-redundant clauses are clauses for which it can be checked efficiently that their addition preserves satisfiability, and (2) every resolvent is a propagation-redundant clause but not vice versa.

The key to constructing short clausal proofs is to aim for deriving short clauses to quickly obtain the empty clause. One approach to achieve this is the *clause learning* technique based on *first unique implication points* [9] used in modern SAT solvers. Starting from a falsifying assignment (found by the solver), this technique derives a redundant clause by computing a subassignment of the falsifying assignment. The derived clause is then the maximal clause that is falsified by that subassignment, thereby ruling out the subassignment and all its extensions. Thus, if the subassignment makes x true and y false, then the derived clause is $\overline{x} \vee y$. When the empty clause is eventually derived, all possible assignments are ruled out, implying that the formula is unsatisfiable.

On mutilated chessboards, clauses intuitively rule out possible placements of dominos. For instance, if a SAT solver arrives at the falsifying placement of the 30 dominos shown on the middle of Fig. 1, it is able to derive the clause that rules out the placement of 14 dominos on the right of Fig. 1, resulting in a 14 literal redundant clause. Deriving this clause rules out all placements that extend the smaller placement, including the falsifying placement of 30 dominos.

Can we immediately learn way shorter clauses, ruling out way more placements? Yes, but not within the resolution proof system. Within the PR proof system, however, much shorter clauses—consisting of only two literals—can be derived. Placements that are represented by such clauses will be discussed below.

Without Loss of Satisfaction

Consider the placements of dominos in Fig. 2. Although the placement on the left is different from the one on the right, they are equivalent in the sense that both cover exactly the same squares. A common way in mathematics to deal with such similar cases is to argue *without loss of generality*, thereby generalizing a specific

case to other similar cases. Within the PR proof system, we can formalize such arguments by deriving clauses that rule out cases that are similar to others.

For mutilated chessboard problems, PR allows us to derive clauses that rule out placements where two horizontal dominos are placed below each other (like the left two dominos on the left mutilated chessboard of Fig. 2). The reasoning is as follows: If it were possible to extend such a placement to a valid placement that covers the whole board, then this would also be possible for the similar placement where two vertical dominos are placed next to each other (such as the left pattern on the right chessboard of Fig. 2). We thus argue *without loss of satisfaction*: If there was a satisfying assignment before ruling out a placement, then there will be a satisfying assignment afterwards. Ruling out all the placements where two horizontal dominos are placed below each other shrinks the number of possible placements aggressively, thus leading to short proofs.

We observed that this pattern and others can be ruled out within the PR proof system when we analyzed automatically generated PR proofs produced by a modified version of the SAT solver Lingeling. This modified version of Lingeling is based on our recently introduced *satisfaction-driven clause learning* (SDCL) paradigm [7]. In the proofs, derived clauses represent the ruled out assignments while the witnesses represent the equivalent placements. Our modified Lingeling is not particularly strong on the mutilated chessboard problems but these binary clauses stood out since they are so short. We thus believe that the solver might be able to solve large mutilated chessboard problems efficiently if we can equip it with the right decision heuristics.

Another pattern that can be ruled out in the PR proof system is the placement of a horizontal domino on top of two vertical dominos as in the right pattern on the left mutilated chessboard of Fig. 2. Such a pattern can be replaced by moving the two vertical dominos one square up and the horizontal domino two squares down as in the right pattern on the right mutilated chessboard of Fig. 2.

Deriving clauses to rule out both patterns—no two horizontal dominos and no horizontal domino on top of two vertical dominos—on all positions of the mutilated chessboard exponentially reduces the number of placements that a solver explores. We require $\mathcal{O}(n^3)$ PR clauses to rule out these patterns for a $n \times n$ mutilated chessboard. The resulting formula can be easily solved using a usual SAT solver. The observed runtime and number of conflicts is also $\mathcal{O}(n^3)$.

Proof Production and Validation

We constructed and validated PR proofs of reasonably large mutilated chessboard problems. Both the problem encodings and the proofs are available at https://github.com/marijnheule/mchess. The proofs consist of a first part that eliminates the earlier mentioned patterns by deriving PR clauses and a second part that refutes the remaining cases using resolution. We generated the first part of the proofs with a dedicated tool that enumerates the required PR clauses. For the second part, we used a SAT solver. Both proof parts are roughly equal in size. The largest problem instance for which we produced a proof is a 50×50

Table 1. Overview of the proof validation results. The second and third column show the numbers of variables and clauses in the encodings of mutilated chessboard problems. The fourth and fifth column show the numbers of clause addition steps in the PR and DRAT proofs, respectively. The last four columns show the runtimes (in CPU seconds, 2.9 GHz Intel Core i7) of non-verified PR proof checking, PR to DRAT conversion, DRAT proof optimization, and verified DRAT proof checking (certification), respectively.

size	#var	#cls	#PR	#DRAT	check	convert	optimize	certify
20×20	760	2 552	7 598	501 766	0.71	0.99	7.06	10.78
30×30	1 740	5 932	22 879	2 489 657	5.74	11.11	85.38	99.45
40×40	3 120	10 712	48 967	7 776 380	32.77	62.57	488.40	518.38
50×50	4 900	16 892	91 665	18 845 988	134.24	252.01	1 862.03	1 702.61

chessboard. Proof production took only a second. Recall that not even the 20×20 mutilated chessboard problem could be solved in the SAT Competition 2018.

To increase the confidence in the correctness of the proofs, we converted the PR proofs into DRAT proofs for which formally-verified checkers exist. We used the tool `pr2drat` [4] for the conversion, optimized the DRAT proofs using the `drat-trim` tool [12] and validated the optimized proofs using the formally-verified tool `ACL2check` [5]. Table 1 provides an overview of the results. Notice that there is a significant gap between verified and non-verified proof checking. This gap is mainly caused by the blowup of the proofs during the conversion.

Conclusion and Challenges

We constructed and validated short propositional proofs of mutilated chessboard problems in the PR proof system. Our proofs show the unsatisfiability of problem instances that are much larger than the largest instances that can be solved by state-of-the-art SAT solvers. The proofs are based on an argument we found when analyzing automatically generated PR proofs. This argument allows us to rule out two small patterns, which exponentially reduces the number of placements that need to be explored. There is an enormous gap between the size of the proofs generated by the modified Lingeling solver and the ones we constructed manually. We believe that an SDCL solver should be able to produce proofs that are close in size to our manual proofs when using the right heuristics and restart strategy, which we consider an important challenge for future research.

Even though the usage of PR clauses in mutilated chessboard problems goes beyond plain symmetry reasoning—existing symmetry-breaking techniques, both static [2] and dynamic [11], are not effective on these formulas—the general argument has a symmetry-reasoning flavor. To further illustrate the power of the PR proof system, we are seeking examples where PR clauses give an exponential benefit without this kind of global and semantic symmetry argument.

Acknowledgements. The authors thank Alexey Porkhunov for contributing the mutilated chessboard formulas to the 2018 SAT Competition and for his suggestion to study these formulas in the context of the PR proof system, and also thank Jasmin Blanchette for his comments on an earlier version of this paper.

References

1. Alekhnovich, M.: Mutilated chessboard problem is exponentially hard for resolution. Theoretical Computer Science 310(1-3), 513–525 (2004)
2. Aloul, F.A., Markov, I.L., Sakallah, K.A.: Shatter: Efficient symmetry-breaking for boolean satisfiability. In: Proceedings of the 40th Annual Design Automation Conference. pp. 836–839. DAC '03, ACM (2003)
3. Dantchev, S.S., Riis, S.: "Planar" tautologies hard for resolution. In: Proc. of the 42nd Annual Symposium on Foundations of Computer Science (FOCS 2001). pp. 220–229. IEEE Computer Society (2001)
4. Heule, M.J.H., Biere, A.: What a difference a variable makes. In: Proc. of the 24th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2018). LNCS, vol. 10806, pp. 75–92. Springer (2018)
5. Heule, M.J.H., Hunt Jr., W.A., Kaufmann, M., Wetzler, N.D.: Efficient, verified checking of propositional proofs. In: Proc. of the 8th Int. Conference on Interactive Theorem Proving (ITP 2017). LNCS, vol. 10499, pp. 269–284. Springer (2017)
6. Heule, M.J.H., Kiesl, B., Biere, A.: Short proofs without new variables. In: Proc. of the 26th Int. Conference on Automated Deduction (CADE-26). LNCS, vol. 10395, pp. 130–147. Springer (2017)
7. Heule, M.J.H., Kiesl, B., Seidl, M., Biere, A.: PRuning through satisfaction. In: Proc. of the 13th Haifa Verification Conference (HVC 2017). LNCS, vol. 10629, pp. 179–194. Springer (2017)
8. de Klerk, E., van Maaren, H., Warners, J.P.: Relaxations of the satisfiability problem using semidefinite programming. Journal of Automated Reasoning 24(1), 37–65 (2000)
9. Marques-Silva, J.P., Sakallah, K.A.: GRASP: A search algorithm for propositional satisfiability. IEEE Transactions on Computers 48(5), 506–521 (1999)
10. McCarthy, J.: A tough nut for proof procedures. Stanford Artificial Intelligence Project Memo 16 (1964)
11. Metin, H., Baarir, S., Colange, M., Kordon, F.: Cdclsym: Introducing effective symmetry breaking in SAT solving. In: Beyer, D., Huisman, M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 99–114. Springer International Publishing (2018)
12. Wetzler, N.D., Heule, M.J.H., Hunt Jr., W.A.: DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In: Proc. of the 17th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2014). LNCS, vol. 8561, pp. 422–429. Springer (2014)