Remark 4. We note that every clause of $G_m(Y)$ contains the pure literal u. Therefore if pure literal elimination were run during preprocessing by a CDCL solver then it would simply remove the clauses of $G_m(Y)$. However, it is not hard to modify $G_m(Y)$ by appending extra clauses H obtained by adding $\overline{u}$ to an arbitrary satisfiable formula on a new small set of extra variables that does not contain any pure literals. The pure literal preprocessing would then not prune the clauses of $G_m(Y)$ and the initial branch setting $u = 0$ would immediately satisfy these extra clauses H which would eliminate their impact on the CDCL execution in any of our arguments.

Simulating Resolution Without Restarts

We begin with the observation that if a given refutation P is too large, the simulation can simply discard it and instead start with an obvious tree-like resolution refutation with at most $2^n - 1$ derived clauses.

Lemma 2. *Let P be a given resolution refutation of $F_n(X)$. If $length(P) > (2^n - 1)/n^2$, then there is a non-restarting CDCL solver that, even without employing clause learning, when run on $F_n(X)$ produces a tree-like resolution refutation after fewer than $n^2 \cdot length(P)$ branching decisions.*

Proof. Let S be a non-restarting CDCL solver that simply discards P and instead simulates an obvious tree-like resolution refutation P' (w.r.t. any fixed variable order) of $F_n(X)$. P' essentially enumerates the 2^n possible variable assignments σ and identifies a clause of $F_n(X)$ (or a learned clause, if S uses clause learning) falsified by each such σ. P' will need no more than $2 \cdot (2^{n-1} - 1) = 2^n - 2$ branching decisions as the $(n-1)$-st assignment for any σ will either satisfy all clauses of $F_n(X)$ or unit propagate the remaining variable. By the precondition on the length of P, the number of branching decisions is thus smaller than $n^2 \cdot length(P)$. □

In the rest of this section, we will thus assume w.l.o.g. that $length(P) \leq (2^n - 1)/n^2$. Pipatsrisawat and Darwiche (2011) provide a CDCL simulation of P, henceforth referred to as the *PD simulation*, that uses $R \leq n^2 \cdot length(P)$ restarts and has size $\mathrm{O}(n^4 \cdot length(P))$. By the above observation regarding $length(P)$, we have $R \leq (2^n - 1)$. Applying Theorem 1 using a counter with $m = n$ bits thus yields the following:

Corollary 1. *Let P be a resolution refutation of $F_n(X)$. Then there is a non-restarting CDCL solver that infers any one asserting clause per conflict when run on formula $F_n(X) \wedge G_n(Y)$ produces a resolution refutation of $F_n(X)$ of size $\mathrm{O}(n^4 \cdot length(P))$.*

We can strengthen this by using a variant of the PD simulation together with a somewhat sharpened analysis.

Theorem 2. *Let P be a resolution refutation of $F_n(X)$. Then a resolution refutation of $F_n(X)$ of length $\mathrm{O}(n^2 \cdot size(P))$ may be found using only $\mathrm{O}(n^2 \cdot size(P))$ variable assignments by a non-restarting CDCL solver that infers any one asserting clause per conflict when run on formula $F_n(X) \wedge G_n(Y)$.*

Algorithm 1: p-simulation of a resolution refutation by a CDCL solver with simple preprocessing

Input : CNF formula $F_n(X)$ with a resolution refutation $P = (C_1, \ldots, C_s)$
Output: A resolution refutation Q of $F_n(X)$

1 **begin**
2 $F' \leftarrow F_n(X) \wedge G_n(Y)$
3 $Q \leftarrow$ a sequence with all clauses of $F_n(X)$
4 $J \leftarrow 0$
5 **for** k *in* $1..s$ **do**
6 **if** C_k *is absorbed by* F' **then**
7 continue
8 Continue CDCL execution on the Y variables as in Lemma 1 until y_0 is assigned a value
9 Let d be the current decision level
10 **while** C_k *is not absorbed by* F' **do**
11 Let $\ell \in C_k$ be a literal witnessing that C_k is not absorbed by F'
12 **repeat**
13 Branch on $\overline{\ell'}$ for all $\ell' \in C_k \setminus \{\ell\}$
14 Branch on $\overline{\ell}$ at decision level $d + |C_k|$
15 Realize a conflict involving only X
16 Derive an asserting clause C over X
17 Append to Q resolution proof of C from F' given by the conflict graph
18 $F' \leftarrow F' \wedge C$
19 $b \leftarrow$ asserting level of C (0 or $> d$)
20 Backjump to decision level b
21 **until** C_k *is absorbed by* F' *w.r.t.* ℓ
22 **if** $b > d$ **then**
23 Branch on $\overline{v}$ at level $b + 1$ (continues execution on $G_n(Y)$ as in Lemma 1)
24 Learn clause 1-equivalent to D_{2J+1} with asserting level d
25 Backtrack to decision level d
26 Learn clause 1-equivalent to D_{2J+2} with asserting level $d - 1$
27 Backtrack to decision level $d - 1$
28 $J \leftarrow J + 1$
29 Backtrack to decision level 0; Realize a conflict
30 **return** Q
31 **end**

Proof. If $length(P) > (2^n - 1)/n^2$, the result follows from Lemma 2. Otherwise we proceed as follows. We assume w.l.o.g. that P is minimal. We observe that since $G_n(Y)$ is satisfiable and on disjoint variables from $F_n(X)$, any minimal resolution refutation of $F_n(X) \wedge G_n(Y)$ must be a refutation of $F_n(X)$.

The CDCL solver execution is given by Algorithm 1. The simulation begins (Line 2) by setting F' to $F_n(X) \wedge G_n(Y)$. The idea, following the PD simulation, is to have F' absorb each of the inferred clauses one by one. When the outer for-loop (Line 5) of the algorithm finishes, F' must have absorbed the empty clause and hence produced a resolution

refutation of $F_n(X) \wedge G_n(Y)$ and thus of $F_n(X)$.

We start (Line 8) by branching on the Y variables as in Lemma 1 until y_0 is assigned a value at some decision level d. In order to absorb a clause C_k that is currently not absorbed by F', the simulation identifies a literal $\ell \in C$ witnessing that C_k is not yet absorbed (Line 11) and then branches negatively on all literals of C_k other than ℓ (Line 13). Since C is not absorbed, making these $|C_k| - 1$ branching decisions will be possible in spite of eager unit propagation, will not cause a conflict by unit propagation, and will allow branching on $\overline{\ell}$. This last branch (Line 14), however, *will* cause a conflict because $C_1, \ldots C_{k-1}$ are already absorbed by F'. From this conflict, S will learn an asserting clause C (Line 16) whose resolution derivation involves ℓ. Since these clauses are only derived from $F_n(X)$, C must be a clause over X. If the asserted literal in C is different from ℓ and C_k remains unabsorbed w.r.t. ℓ, this process of branching on $\overline{\ell}$ is repeated (Lines 12-21) no more than $n - |C_k| + 1$ times until ℓ becomes the asserted literal (this holds because all asserted variables here must be distinct and cannot belong to $C_k \setminus \{\ell\}$). S will backtrack to some asserting level b (Line 19), which must either be the level of some X variable or be 0 (if S learns a unit clause). At this point, we have absorbed C_k w.r.t. ℓ. If $b > d$, however, the "leftover" branching decisions on the X variables of C_k may interfere with the derivation of clauses needed to absorb C_k w.r.t. other literals or to absorb $C_{k+1}, \ldots C_s$. In order to avoid this situation, the PD simulation simply restarts the solver and forcibly returns to decision level 0. Since S is a non-restarting CDCL solver, our construction employs the counter involving the Y variables and a decision setting $v = 0$ (Line 23) to backtrack over any leftover branching decisions on the X variables that remain after absorbing C_k. This increments the counter and brings us back to decision level $d - 1$ (Lines 24-27).

Repeating this process (Lines 10-28) for other literals of C_k witnessing that C_k is not yet absorbed will eventually result in C_k being absorbed. Thus, after no more than $(n - |C_k| + 1)|C_k|$ repetitions, each of which involves at most $|C_k|$ branching decisions or at most n resolution steps, we are able to absorb C_k. This yields a total of at most $n \sum_{k=1}^{s} |C_k|^2$ branching decisions and at most $n^2 \sum_{k=1}^{s} |C_k| = n^2 \cdot size(P)$ resolution steps, to absorb all clauses of P.

The total number R of restarts needed to be simulated is at most $n^2 \cdot length(P)$ which, by our earlier argument, is at most $2^n - 1$. Hence, $\log_2(R+1) \leq n$ and, from Theorem 1, it suffices to use $G_n(Y)$ as the counter for $F_n(X)$. □

Concluding Remarks

Known efficient simulations of resolution refutations by modern CDCL SAT solvers that learn one asserting clause per conflict have so far relied heavily on the ability of such solvers to restart several times per resolution derivation step that they are trying to simulate. Our result shows that simple preprocessing and an associated branching heuristic can replace the need for these solvers to explicitly restart. As a consequence, from a theoretical standpoint, such solvers are in fact powerful enough to efficiently simulate resolution refutations without relying on restarts at all.

The result holds for all CDCL solvers that learn (at least) one asserting clause from each conflict, irrespective of whether or not they employ other common techniques such as backjumping and phase saving. As remarked briefly earlier, the construction can be extended to withstand simplification techniques such as pure literal elimination.

The kind of preprocessing used in our non-restarting simulation is admittedly of a rather different nature than the commonly used preprocessing techniques geared towards increasing the efficiency of SAT solvers. It adds a counting formula on a brand new set of variables such that branching first on (a subset of) these new variables enables the solver to freely backtrack out of any "leftover" partial assignment after realizing a conflict.

While our simulation result answers an open theoretical question, research on effective restart strategies continues to be relevant for CDCL solvers in practice. The question of whether modern CDCL solvers can polynomially simulate resolution proofs without restarting and without preprocessing the input formula remains open.

Acknowledgments

The work of the first author was supported by NSF grant CCF-1217099. Part of this research was conducted when the authors were visiting the BIRS Workshop on Theoretical Foundations of Applied SAT Solving during January 2014.

References

Atserias, A.; Fichte, J. K.; and Thurley, M. 2009. Clause-learning algorithms with many restarts and bounded-width resolution. In *12th SAT*, volume 5584 of *LNCS*, 114–127.

Atserias, A.; Fichte, J. K.; and Thurley, M. 2011. Clause-learning algorithms with many restarts and bounded-width resolution. *JAIR* 40:353–373.

Audemard, G., and Simon, L. 2012. Refining restarts strategies for SAT and UNSAT. In *18th CP*, volume 7514 of *LNCS*, 118–126.

Beame, P.; Kautz, H.; and Sabharwal, A. 2003. Understanding the power of clause learning. In *18th IJCAI*, 1194–1201.

Beame, P.; Kautz, H.; and Sabharwal, A. 2004. Understanding and harnessing the potential of clause learning. *JAIR* 22:319–351.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2009. *Handbook of Satisfiability*. IOS Press.

Buss, S. R., and Bonet, M. L. 2012. An improved separation of regular resolution from pool resolution and clause learning. In *15th SAT*, volume 7313 of *LNCS*, 244–57.

Buss, S. R., and Kolodziejczyk, L. 2012. Small stone in pool. Manuscript, 2012.

Buss, S. R.; Hoffmann, J.; and Johannsen, J. 2008. Resolution trees with lemmas: Resolution renements that characterize DLL-algorithms with clause learning. *Logical Methods in Computer Science* 4(4):13.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *AI J.* 32(1):97–130.

Frost, D.; Rish, I.; and Vila, L. 1997. Summarizing csp hardness with continuous probability distributions. In *14th AAAI*, 327–333.

Gomes, C. P.; Selman, B.; and Crato, N. 1997. Heavy-tailed distributions in combinatorial search. In *3rd CP*, volume 1330 of *LNCS*, 121–135.

Gomes, C. P.; Selman, B.; and Kautz, H. 1998. Boosting combinatorial search through randomization. In *15th AAAI*, 431–437.

Gray, F. 1953. Pulse code communication. US Patent #2632058, (filed 1947).

Haim, S., and Walsh, T. 2009. Restart strategy selection using machine learning techniques. In *12th SAT*, volume 5584 of *LNCS*, 312–325.

Hertel, P.; Bacchus, F.; Pitassi, T.; and Van Gelder, A. 2008. Clause learning can effectively p-simulate general propositional resolution. In *23rd AAAI*, 283–290.

Hogg, T., and Williams, C. P. 1994. Expected gains from parallelizing constraint solving for hard problems. In *12th AAAI*, 331–336.

Luby, M.; Sinclair, A.; and Zuckerman, D. 1993. Optimal speedup of Las Vegas algorithms. *Inf. Process. Lett.* 47(4):173–180.

Marques-Silva, J. P., and Sakallah, K. A. 1996. GRASP – a new search algorithm for satisfiability. In *ICCAD*, 220–227.

Marques-Silva, J. P.; Lynce, I.; and Malik, S. 2009. CDCL solvers. In Biere et al. (2009). chapter 4, 131–154.

Moskewicz, M. W.; Madigan, C. F.; Zhao, Y.; Zhang, L.; and Malik, S. 2001. Chaff: Engineering an efficient SAT solver. In *38th DAC*, 530–535.

Pipatsrisawat, K., and Darwiche, A. 2009. On the power of clause-learning SAT solvers with restarts. In *15th CP*, volume 5732 of *LNCS*, 654–668.

Pipatsrisawat, K., and Darwiche, A. 2011. On the power of clause-learning SAT solvers as resolution engines. *AI J.* 175(2):512–525.

Robinson, J. A. 1965. A machine-oriented logic based on the resolution principle. *J. Assoc. Comput. Mach.* 12(1):23–41.

Stallman, R. M., and Sussman, G. J. 1977. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *AI J.* 9:135–196.

Van Gelder, A. 2005. Pool resolution and its relation to regular resolution and DPLL with clause learning. In *12th Intl. Conf. Logic for Prog., AI, and Reason.*, volume 3835 of *LNCS*, 580–594.

Walsh, T. 1999. Search in a small world. In *16th IJCAI*, 1172–1177.

Zhang, L.; Madigan, C. F.; Moskewicz, M. H.; and Malik, S. 2001. Efficient conflict driven learning in a Boolean satisfiability solver. In *ICCAD*, 279–285.