Pool resolution is NP-hard to recognize

Sam Buss*
Department of Mathematics
University of California, San Diego
La Jolla, CA 92093-0112, USA
sbuss@math.ucsd.edu

September 10, 2009

Abstract

A pool resolution proof is a dag-like resolution proof which admits a depth-first traversal tree in which no variable is used as a resolution variable twice on any branch. The problem of determining whether a given dag-like resolution proof is a valid pool resolution proof is shown to be NP-complete.

Propositional resolution has been the foundational method for reasoning in propositional logic, especially for forming refutations of satisfiability of set of clauses. In recent years, the most successful satisfiability testers have used the DPLL (Davis-Putnam-Logeman-Loveland) algorithm combined with clause learning, backtracking, restarts, and other techniques. (See Beame, Kautz and Sabharwal [4] for an overview of clause learning.) Pool resolution was introduced by Van Gelder [11] as an resolution-based refutation system that provides a good theoretical model for the proofs produced by real-world satisfiability testing algorithms that incorporate clause learning and backtracking. Van Gelder proved that pool resolution is exponentially stronger than regular resolution. Bacchus, Hertel, Pitassi and Van Gelder [3], building on techniques from [4], proved that pool resolution can "effectively p-simulate" full resolution; and Buss, Hoffmann, and Johannsen [6, Th. 19] gave an effective p-simulation for a system similar to pool resolution. However, it is open whether pool resolution can directly p-simulate full resolution.

*Supported in part by NSF grant DMS-0700533.

Van Gelder defined pool resolution algorithmically; however, we shall use his characterization that a pool resolution proof is a dag-like resolution proof that admits a regular, depth-first traversal. A depth-first traversal defines a tree on the clauses in the proof, which is a subgraph of the dag. The tree is called "regular", provided that no branch in the tree that contains two clauses that are derived by resolution on the same variable.

Actually, Van Gelder defined pool resolution using an extended form of the resolution that allows any two clauses to be resolved with any resolution variable — regardless of whether the variable occurs appropriately in the clauses. This extended resolution rule was called the *degenerate* resolution rule by [3].

A depth first traversal τ of a refutation R and the associated traversal tree T_τ are formally defined as follows. If C is a non-initial clause in R and D is one of the hypotheses of the inference used to derive C, then we call D a *child* of C. We assume w.l.o.g. that R is rooted, that is, that every clause in R is a descendent of the empty clause. A depth first traversal τ of R is a sequence $E_0, E_1, \ldots, E_p$ containing the clauses of R, each clause exactly once, starting with the empty clause. For $1 \le i \le m$, E_i must be a child of an earlier E_j, where j must be the maximum value $< i$ such that not all of E_j's children occur among $E_0, \ldots, E_{i-1}$. In this case, E_i is also a child of E_j in the tree T_τ induced by the traversal τ, and all edges in T_τ are obtained in this way.

The traversal τ is called *regular* provided T_τ has no branch that contains two clauses derived by resolution on the same variable. R is a pool resolution refutation if and only if it admits a regular depth first traversal.

The Pool Resolution problem is the decision problem of deciding whether a given dag-like resolution proof R is also a pool resolution refutation. Note that this problem is clearly in NP, since the algorithm can just non-deterministically guess a regular, depth-first traversal.

Theorem 1 *The* Pool Resolution *problem is NP-complete.*

To fully specify the Pool Resolution problem, we need to say how the dag-like proof R is presented. Our proof of Theorem 1 will make the strongest possible assumptions: First, we will work only with proofs R that are refutations in which all resolution inferences are standard. (A "refutation" is a proof that ends with the contradictory clause $\emptyset$.) Furthermore, the refutation R will be specified as a sequence of clauses, and each non-initial clause can be derived in exactly one way from the earlier clauses. Thus, R will admit a unique dag structure.

There have been a number of results, including [1, 2, 8, 9, 10], about the hardness of finding resolution proofs, or of determining whether resolution proofs exist. Theorem 1, however, is more in the spirit of hardness results by Buss-Hoffmann [5] and Hoffmann [7]: these show that, given a particular resolution refutation, it is hard to determine if it satisfies extra conditions.

The rest of the paper gives the proof of the theorem. The main construction for the proof will be a reduction from the NP-complete satisfiability problem SAT to POOL RESOLUTION. An instance Γ of SAT consists of a set of m clauses $C_1, \ldots, C_m$ involving k variables $x_1, \ldots, x_k$.

Given Γ, we will construct another set Π of clauses and a dag-like resolution refutation R of Π. The propositional variables in Π will be u_i and v_i for $1 \leq i \leq k$, c_j for $1 \leq j \leq m$, and one further variable y. We will prove that R is a valid pool resolution refutation iff Γ is satisfiable.

The root portion of the refutation R is shown in Figure 1. The figure uses the following conventions. (1) Each node in the dag is labeled with a clause. (2) Each non-initial clause C has two children (immediate successors) D_0 and D_1, indicated by edges drawn from C upward towards D_0 and D_1, such that C is inferred from the two children clauses using resolution with respect to some *resolution variable*. (3) The resolution variable is easily determined from D_0 and D_1, and is also indicated in the column on the right side of the figure. (4) Initial clauses are written in boldface. (5) Other leaves in the figure, decorated with $\cdot\cdot\vdots\cdot\cdot$'s are *not* initial clauses; rather their derivations are shown in other figures.

The remaining portions of R are shown in in Figures 2-4. It should be noted that no clause appears more than once in R. In particular, the clauses c_i are used multiple times in Figures 2 and 3, but these represent multiple uses of the same clause, and each c_i is derived exactly once as shown in Figure 4.

Examining the refutation R in Figures 1-4 shows that the only way that a traversal τ can fail to be regular is for the resolution variable y to be used twice along some branch of T_τ. In fact, the variable y is the only variable that is used twice along any directed path in R.

As shown in Figure 4, the variable y is the resolution variable used to derive each clause c_j. It is also used as the resolution variable at the top of Figure 1. In the traversal tree T_τ, the clause yv_k will be the child of the clause $v_1v_2\cdots v_k$ which is derived using y as the resolution variable. In addition, as shown in Figure 2, c_j is in the sub-derivation of R rooted at yv_k. Therefore, if there is any clause c_j which is not visited before yv_k in the traversal, then there will be a branch in T_τ containing two uses of y as a

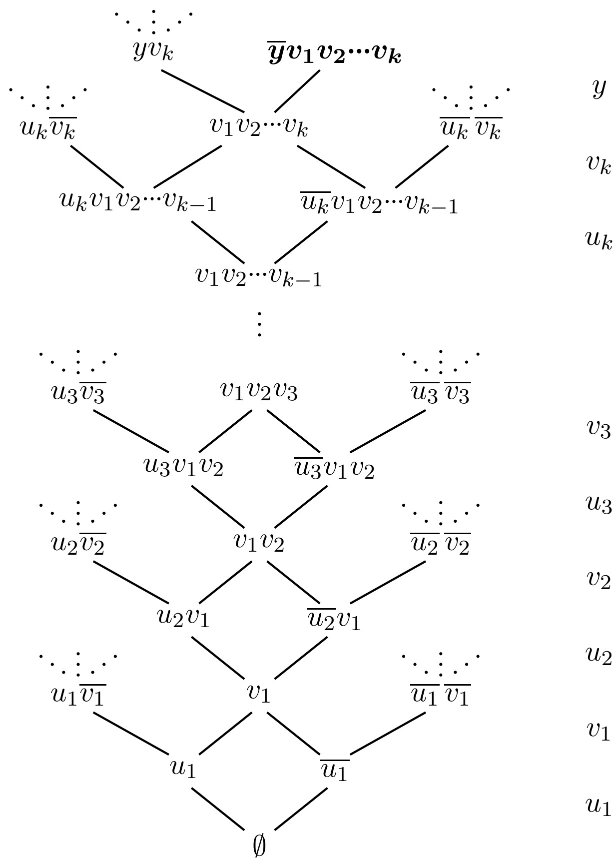

Figure 1: This shows the root portion of the dag refuation R. The end clause is $\emptyset$. The only initial clause, shown in boldface, is $\overline{y}v_1v_2\cdots v_k$. The other leaves, decorated with $\ddots\vdots\cdot^{\cdot^{\cdot}}$'s are derived from the proof fragments shown in Figures 2-4. The variables in the right column indicate the resolution variable for the corresponding inferences.

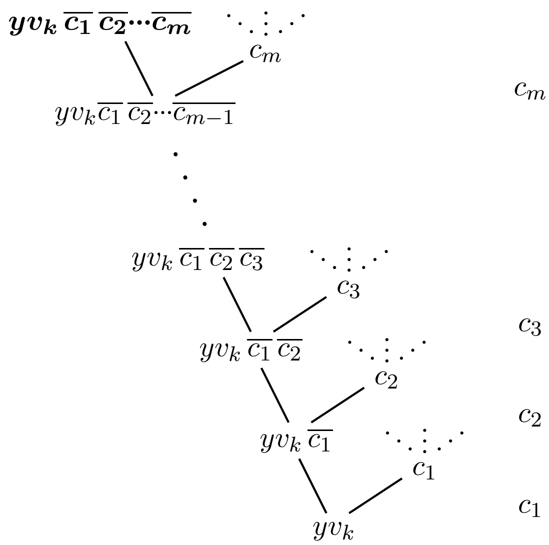

Figure 2: The derivation of the clause yv_k.

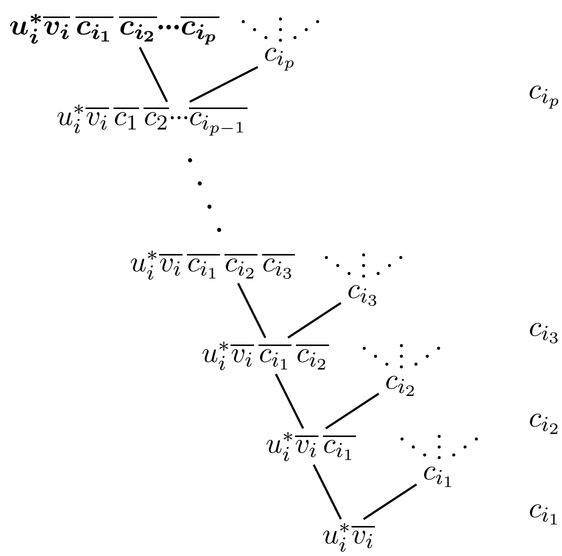

Figure 3: This shows the derivation of $u_i^*\overline{v_i}$, where u_i^* is either u_i or $\overline{u_i}$. Letting ℓ be x_i or $\overline{x_i}$, respectively, then $C_{i_1}, C_{i_2}, \ldots, C_{i_p}$ are the clauses that contain ℓ.

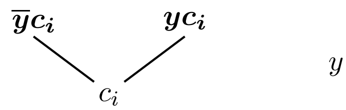

Figure 4: The derivation of c_i.

resolution variable. It follows that any regular traversal must visit every c_j before visiting yv_k.

The only way to visit a clause c_j before yv_k is by visiting the clauses $u_i^*\overline{v_i}$ that are derived as shown in Figure 3, where u^* is either u_i or $\overline{u_i}$. There are $2k$ such sub-derivations, two for each Γ-variable x_i. Fixing the value of i, let the literal ℓ be either x_i or $\overline{x_i}$. In the first case, the variable u_i^* is u_i, and in the second case, u_i^* is $\overline{u_i}$. Let $\mathcal{C}(\ell)$ be the set of clauses in Γ which contain ℓ, and enumerate this set as $\mathcal{C}(\ell) = \{C_{i_1}, \ldots, C_{i_p}\}$. Here $p = p(\ell)$ is the number of clauses that contain ℓ. Then, the clause $u_i^*\overline{v_i}$ is derived as shown in Figure 3. Note in particular, that the derivation of $u_i^*\overline{v_i}$ includes the derivations of the clauses $c_{i_1}, \ldots, c_{i_p}$.

Lemma 2 *Let τ be a depth-first traversal of R and $1 \leq i \leq k$. Then at most one of the clauses $u_i\overline{v_i}$ and $\overline{u_i}\,\overline{v_i}$ can appear in τ before the clause yv_k.*

The proof of the lemma is almost obvious. Suppose $u_i\overline{v_i}$ appears in the traversal before $\overline{u_i}\ \overline{v_i}$. This means that $u_iv_1 \cdots v_{i-1}$ also appears in the traversal before $\overline{u_i}\,\overline{v_i}$. Hence, since yv_k is in the sub-derivation rooted at $u_iv_1 \cdots v_{i-1}$ and $\overline{u_i}\,\overline{v_i}$ is not, it follows that yv_k precedes $\overline{u_i}\,\overline{v_i}$ in the traversal. A similar argument applies if $\overline{u_i}\,\overline{v_i}$ precedes $u_i\overline{v_i}$ in the traversal. □

We define a partial truth assignment α_τ as follows.

$$\alpha_\tau(x_i) \;=\; \begin{cases} T & \text{if } u_i\overline{v_i} \text{ precedes } yv_k \text{ in } \tau \\ F & \text{if } \overline{u_i}\,\overline{v_i} \text{ precedes } yv_k \text{ in } \tau \\ * & \text{otherwise} \end{cases}$$

where T, F, and $*$ represent the values *True*, *False*, and "undefined". The third situation arises when neither clause precedes yv_k in τ. The partial assignment α_τ induces a (partial) truth assignment on literals in the obvious way, and α_τ satisfies Γ provided every $C_i \in \Gamma$ contains at least one literal that is set to *True* by α_τ.

Lemma 3 *The traversal τ is regular if and only if α_τ satisfies* Γ.

To prove the lemma, first suppose α_τ satisfies Γ. Then each clause C_j in Γ contains some literal ℓ such that $\alpha_\tau(\ell) = T$. Letting, u_i^* equal u_i or $\overline{u_i}$, respectively, if ℓ is x_i or $\overline{x_i}$, this means $u_i^*\overline{v_i}$ is traversed in τ before yv_k. Therefore, since C_j is one of the clauses containing ℓ, the unit clause c_j is also traversed before yv_k.

It follows, that if α_τ satisfies Γ, then every c_j is traversed before yv_k. This suffices to make the traversal τ regular.

Now suppose α_τ does not satisfy Γ. Let C_j be a clause in Γ that is not made true by α_τ. By Lemma 2, this means that there is no $u_i^*\overline{v_i}$ which is traversed before yv_k which has the unit clause c_j in its sub-derivation. Therefore, c_j is traversed after yv_k. This ensures that τ is not a regular traversal since y is used as a resolution variable both to derive the clause $v_1v_2\cdots v_k$ from yv_k, and to derive c_j, and since c_j is in the sub-derivation rooted at yv_k. □

Lemma 3 shows that if R has a regular traversal, then Γ is satisfiable. On the other hand, if α is a satisfying assignment for Γ, then it is straightforward to construct a traversal τ such that $\alpha_\tau = \alpha$.

That completes the proof of the theorem.

Acknowledgements. I thank Philip Hertel, Jan Hoffmann, Toni Pitassi, and Allen Van Gelder for useful discussions on this problem, and the referee for further comments.

References

[1] M. Alekhnovich, S. Buss, S. Moran, and T. Pitassi, *Minimum propositional proof length is NP-hard to linearly approximate*, Journal of Symbolic Logic, 66 (2001), pp. 171–191. A shorter extended abstract appeared in *Mathematical Foundations of Computer Science (MFCS'98)*, Springer-Verlag Lecture Notes in Computer Science #1450, 1998, pp. 176-184.

[2] M. Alekhnovich and A. A. Razborov, *Resolution is not automatizable unless $W[P]$ is tractable*, in Proc. 42nd IEEE Conf. on Foundations of Computer Science (FOCS), 2001, pp. 210–219.

[3] F. Bacchus, P. Hertel, T. Pitassi, and A. Van Gelder, *Clause learning can effectively p-simulate general propositional resolution*, in Proc. 23rd AAAI Conf. on Artificial Intelligence (AAAI 2008), AAAI Press, 2008, pp. 283–290.

[4] P. Beame, H. A. Kautz, and A. Sabharwal, *Towards understanding and harnessing the potential of clause learning*, J. Artificial Intelligence Research, 22 (2004), pp. 319–351.

[5] S. R. Buss and J. Hoffmann, *The NP-hardness of finding a directed acyclic graph for regular resolution*, Theoretical Computer Science, 396 (2008), pp. 271–276.

[6] S. R. Buss, J. Hoffmann, and J. Johannsen, *Resolution trees with lemmas: Resolution refinements that characterize DLL-algorithms with clause learning*, Logical Methods of Computer Science, 4 (2008). Issue 4, Article 13.

[7] J. Hoffmann, *Finding a tree structure in a resolution proof is NP-complete*, Theoretical Computer Science, 410 (2009), pp. 2295–2300.

[8] K. Iwama, *Complexity of finding short resolution proofs*, in Mathematical Foundations of Computer Science 1997, I. Prívara and P. Ruzicka, eds., Lecture Notes in Computer Science #1295, Springer-Verlag, 1997, pp. 309–318.

[9] K. Iwama and E. Miyano, *Intractibility of read-once resolution*, in Proceedings of the Tenth Annual Conference on Structure in Complexity Theory, Los Alamitos, California, 1995, IEEE Computer Society, pp. 29–36.

[10] S. Szeider, *NP-completeness of refutability by literal-once resolution*, in Automated Reasoning: First International Joint Conference, (IJCAR), Springer Verlag, 2001, pp. 168–181.

[11] A. Van Gelder, *Pool resolution and its relation to regular resolution and DPLL with clause learning*, in Logic for Programming, Artificial Intelligence, and Reasoning (LPAR), Lecture Notes in Computer Science Intelligence 3835, Springer-Verlag, 2005, pp. 580–594.