An Improved Separation of Regular Resolution from Pool Resolution and Clause Learning (Extended Abstract)

Maria Luisa Bonet* and Sam Buss**

[1] Lenguajes y Sistemas Informáticos, Universidad Politécnica de Cataluña, Barcelona, Spain bonet@lsi.upc.edu
[2] Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112, USA sbuss@math.ucsd.edu

Abstract. We prove that the graph tautology principles of Alekhnovich, Johannsen, Pitassi and Urquhart have polynomial size pool resolution refutations that use only input lemmas as learned clauses and without degenerate resolution inferences. These graph tautology principles can be refuted by polynomial size DPLL proofs with clause learning, even when restricted to greedy, unit-propagating DPLL search.

1 Introduction

DPLL algorithms with clause learning have been highly successful at solving real-world instances of satisfiability (SAT), especially when extended with techniques such as clause learning, restarts, variable selection heuristics, etc. The basic DPLL procedure without clause learning or restarts is equivalent to tree-like resolution. The addition of clause learning makes DPLL considerably stronger. In fact, clause learning together with unlimited restarts is capable of simulating general resolution proofs [12]. However, the exact power of DPLL with clause learning but without restarts is unknown. This question is interesting both for theoretical reasons and for the potential for better understanding the practical performance of DPLL with clause learning.

Beame, Kautz, and Sabharwal [3] gave the first theoretical analysis of DPLL with clause learning. Among other things, they noted that clause learning with restarts simulates full resolution. Their construction required the DPLL algorithm to ignore some contradictions, but this was rectified by Pipatsrisawat and Darwiche [12] who showed that SAT solvers which do not ignore contradictions can also simulate resolution. (See [2] for the bounded width setting.)

[3] also studied DPLL clause learning without restarts. Using "proof trace extensions", they were able to show that DPLL with clause learning and no restarts is strictly stronger than any "natural" proof system strictly weaker than

* Supported in part by grant TIN2010-20967-C04-02.
** Supported in part by NSF grants DMS-0700533 and DMS-1101228, and by a grant from the Simons Foundation (#208717 to Sam Buss)

resolution. Here, a *natural* proof system is one in which proofs do not increase in length when variables are restricted to constants. However, the proof trace method and the improved constructions of [9, 7] have the drawback of introducing extraneous variables and clauses, and using contrived resolution refutations.

There have been two approaches to formalizing DPLL with clause learning as a static proof system rather than as a proof search algorithm. The first approach was pool resolution with a degenerate resolution inference [16, 9]. Pool resolution requires proofs to have a depth-first regular traversal similarly to the search space of a DPLL algorithm. Degenerate resolution allows resolution inferences in which the hypotheses may lack occurrences of the resolution literal. Van Gelder [16] argued that pool resolution with degenerate resolution inferences simulates a wide range of DPLL algorithms with clause learning. He also gave a proof, based on [1], that pool resolution with degenerate inferences is stronger than regular resolution, using extraneous variables similar to proof trace extensions.

The second approach [7] is the proof system regWTRI that uses a "partially degenerate" resolution rule called w-resolution, and clause learning of input lemmas. [7] showed that regWRTI exactly captures non-greedy DPLL with clause learning. By "non-greedy" is meant that contradictions may need to be ignored.

It remains open whether any of DPLL with clause learning, pool resolution, or the regWRTI proof system can polynomially simulate general resolution. One approach to answering these questions is to try to separate pool resolution (say) from general resolution. So far, however, separation results are known only for the weaker system of regular resolution; namely, Alekhnovitch et al. [1], gave an exponential separation between regular resolution and general resolution based on two families of tautologies, variants of the graph tautologies GT' and the "Stone" pebbling tautologies. Urquhart [15] subsequently gave a related separation.[3] In the present paper, we call the tautologies GT' the *guarded* graph tautologies, and henceforth denote them GGT instead of GT'.

The obvious next question is whether pool resolution (say) has polynomial size proofs of the GGT tautologies or the Stone tautologies. The main result of the present paper resolves the first question by showing that pool resolution does indeed have polynomial size proofs of the graph tautologies GGT. Our proofs apply to the original GGT principles, without the use of extraneous variables in the style of proof trace extensions; and our refutations use only the traditional resolution rule, not degenerate resolution inferences or w-resolution inferences. In addition, we use only learning of input clauses; thus, our refutations are also regWRTI proofs (and in fact regRTI proofs) in the terminology of [7]. As a corollary of the characterization of regWRTI by [7], the GGT principles have polynomial size refutations that can found by a DPLL algorithm with clause learning and without restarts (under the appropriate variable selection order).

It is still open if there are polynomial size pool resolution refutations for the Stone principles. A much more ambitious project would be to show that pool

[3] Huang and Yu [10] also gave a separation of regular resolution and general resolution, but only for a single set of clauses. Goerdt [8] gave a quasipolynomial separation of regular resolution and general resolution.

resolution or regWRTI can simulate full resolution, or that DPLL with clause learning and without restarts can simulate full resolution. It is far from clear that this holds, but, if so, our methods may represent a step in that direction.

The first idea for constructing our pool resolution or regRTI proofs might be to try to follow the regular refutations of the graph tautology clauses GT_n as given by [14, 5, 17]: however, these refutations cannot be used directly since the transitivity clauses of GT_n are "guarded" in the GGT_n clauses and this yields refutations which violate the regularity/pool property. So, the second idea is that the proof search process branches as needed to learn transitivity clauses. This generates additional clauses that must be proved: to handle these, we develop a notion of "bipartite partial order" and show that the refutations of [14, 5, 17] can still be used in the presence of a bipartite partial order. The tricky part is to be sure that exactly the right set of clauses is derived by each subproof.

Our refutations of the GGT_n tautologies can be modified so that they are "greedy" and "unit-propagating". This means that, at any point in the proof search process, if it is possible to give an "input" refutation of the current clause, then that refutation is used immediately. The greedy and unit-propagating conditions correspond well to actual implemented DPLL proof search algorithms which backtrack whenever a contradiction can be found by unit propagation (c.f., [9]). The paper concludes with a short description of a greedy, unit-propagating DPLL clause learning algorithm for GGT_n.

For space reasons, only the main constructions for our proofs are included in this extended abstract. Complete proofs are in the full version of the paper available at the authors' web pages and at `http://arxiv.org/abs/1202.2296`.

2 Preliminaries and Main Results

Propositional variables range over the values *True* and *False*. The notation $\overline{x}$ expresses the negation of x. A *literal* is either a variable x or a negated variable $\overline{x}$. A *clause* C is a set of literals, interpreted as the disjunction ($\vee$) of its members.

Definition 1. *The various forms of resolution take two premise clauses A and B and a resolution literal x, and produce a new clause C called the* resolvent.

$$\frac{A \qquad B}{C}$$

It is required that $\overline{x} \notin A$ and $x \notin B$. The different forms of resolution are:

Resolution rule. *Here $A := A' \vee x$ and $B := B' \vee \overline{x}$, and C is $A' \vee B'$.*

Degenerate resolution rule. [9, 16] *If $x \in A$ and $\overline{x} \in B$, we apply the resolution rule to obtain C. If A contains x, and B doesn't contain $\overline{x}$, then the resolvent C is B. If A doesn't contain x, and B contains $\overline{x}$, then the resolvent C is A. If neither A nor B contains the literal x or $\overline{x}$, then C is the lesser of A or B according to some tiebreaking ordering of clauses.*

w-resolution rule. [7] *Here $C := (A \setminus \{x\}) \vee (B \setminus \{\overline{x}\})$. If the literal $x \notin A$ (resp., $\overline{x} \notin B$), then it is called a* phantom literal *of A (resp., B).*

A *resolution derivation* of a clause C from a set F of clauses is a sequence of clauses that derives C from the clauses of F using resolution. Degenerate and w-resolution derivations are defined similarly. A *refutation* of F is a derivation of the empty clause. A refutation is *tree-like* if its underlying graph is a tree. A resolution derivation is *regular* provided that, along any path in the directed acyclic graph, each variable is resolved at most once and provided that no variable appearing in the final clause is used as a resolution variable.

Resolution is well-known to be sound and complete; in particular, C is a consequence of F iff there is a derivation of some $C' \subseteq C$ from F.

We define pool resolution using the conventions of [7], who called this concept "tree-like regular resolution with lemmas". The idea is that any clause appearing in the proof is a learned lemma and can be used freely from then on.

Definition 2. *The* postorder *ordering $<_T$ of the nodes in a tree T is defined so that if u is a node of T, v a node in the subtree rooted at the left child of u, and w a node in the subtree rooted at the right child of u, then $v <_T w <_T u$.*

Definition 3. *A* pool resolution *proof from a set of initial clauses F is a resolution proof tree T that fulfills the following conditions: (a) each leaf is labeled either with a clause of F or with a clause (called a "lemma") that appears earlier in the tree in the $<_T$ ordering; (b) each internal node is labeled with a clause and a literal, and the clause is obtained by resolution from the clauses labeling the node's children, by resolving on the given literal; (c) the proof tree is regular; (d) the root is labeled with the conclusion clause (the empty clause in the case of a pool refutation).*

The notions of *degenerate pool resolution* proof and *pool w-resolution* proof are defined similarly. Note that [16, 9] defined pool resolution to be the degenerate pool resolution system, so our notion of pool resolution is more restrictive than theirs. (Our definition is equivalent to the one in [6], however.)

A "lemma" in part (a) of Definition 3 is called an *input lemma* if it is derived by *input* subderivation, namely by a subderivation in which each inference has at least one hypothesis which is a member of F or is a lemma.

The various graph tautologies, sometimes also called "ordering principles" use a size parameter $n > 1$, and variables $x_{i,j}$ with $i, j \in [n]$ and $i \neq j$, where $[n] = \{0, 1, 2, \ldots, n-1\}$. A variable $x_{i,j}$ will intuitively represent the condition that $i \prec j$ with $\prec$ intended to be a total, linear order. We thus adopt the convention that $x_{i,j}$ and $\overline{x}_{j,i}$ are the identical literal. This identification makes no essential difference to the complexity of proofs of the tautologies, but reduces the number of literals and clauses, and simplifies definitions.

The following tautologies are based on Krishnamurthy [11]. These tautologies, or similar ones, have also been studied by [14, 5, 1, 4, 13, 17].

Definition 4. *Let $n > 1$. Then GT_n is the following set of clauses:*

($\alpha_\emptyset$) The clauses $\bigvee_{j \neq i} x_{j,i}$, for each value $i < n$.
($\gamma_\emptyset$) The transitivity clauses $T_{i,j,k} := \overline{x}_{i,j} \vee \overline{x}_{j,k} \vee \overline{x}_{k,i}$ *for all distinct i, j, k in $[n]$.*

Note that the clauses $T_{i,j,k}$, $T_{j,k,i}$ and $T_{k,i,j}$ are identical.

The next definition is from [1] who used the notation GT'_n. They used particular functions r and s for their lower bound proof, but since our upper bound proof does not depend on the details of r and s we leave them unspecified. We require that $r(i,j,k) \neq s(i,j,k)$ and that the set $\{r(i,j,k), s(i,j,k)\} \not\subset \{i,j,k\}$. W.l.o.g., $r(i,j,k) = r(j,k,i) = r(k,i,j)$, and similarly for s.

Definition 5. *Let $n \geq 1$, and let $r(i,j,k)$ and $s(i,j,k)$ be functions mapping $[n]^3 \to [n]$ as above. The* guarded graph tautology GGT_n *consists of:*

($\alpha_\emptyset$) *The clauses $\bigvee_{j \neq i} x_{j,i}$, for each value $i < n$.*

($\gamma'_\emptyset$) *The* guarded *transitivity clauses $T_{i,j,k} \vee x_{r,s}$ and $T_{i,j,k} \vee \overline{x}_{r,s}$, for all distinct i,j,k in $[n]$, where $r = r(i,j,k)$ and $s = s(i,j,k)$.*

Theorem 1. *The guarded graph tautology principles GGT_n have polynomial size pool resolution refutations.*

Theorem 2. *The guarded graph tautology principles GGT_n have polynomial size, tree-like regular resolution refutations with input lemmas.*

A consequence of Theorem 2 is that the GGT_n clauses can be shown unsatisfiable by non-greedy polynomial size DPLL searches using clause learning. This follows via Theorem 5.6 of [7]. Even better, we can improve the constructions of Theorems 1 and 2 to show that the GGT_n principles can be refuted also by greedy, unit-propagating polynomial size DPLL searches with clause learning.

Definition 6. *Let R be a tree-like regular resolution (or w-resolution) refutation with input lemmas from the initial clauses Γ. Let C be a clause in R. Define $\Gamma(C)$ to be Γ plus every clause $D <_R C$ in R that is derived by an input subproof. Define C^+ to be the set of literals that occur as a literal (or as a literal or phantom literal) in any clause on the path from C down to the root of R.*

The refutation R is greedy *and* unit-propagating *provided that, for each clause C of R, if there is an input derivation from $\Gamma(C)$ of some clause $C' \subseteq C^+$ which does not resolve on any literal in C^+, then C is derived in R by such a derivation.*

Note that, as proved in [3], the condition that there is a input derivation from $\Gamma(C)$ of some $C' \subseteq C^+$ which does not resolve on literals in C^+ is equivalent to the condition that if all literals of C^+ are set false then unit propagation yields a contradiction from $\Gamma(C)$. (In [3], these are called "trivial" proofs.) This justifies the terminology "unit-propagating".

Theorem 3. *The guarded graph tautology principles GGT_n have greedy, unit-propagating, polynomial size, tree-like, regular w-resolution refutations with input lemmas.*

A similar theorem holds for greedy, unit-propagating pool resolution refutations with degenerate resolution inferences.

Theorem 4. *There are DPLL search procedures with clause learning which are greedy, unit-propagating, but do not use restarts, that refute the GGT_n clauses in polynomial time.*

3 Proof of Main Theorems

The following theorem is an important ingredient of our upper bound proof.

Theorem 5. (Stålmarck [14], Bonet-Galesi [5], Van Gelder [17]) *The sets* GT_n *have regular resolution refutations* P_n *of polynomial size* $O(n^3)$.

The refutations P_n can be modified to give refutations of GGT_n by first deriving each transitive clause $T_{i,j,k}$ from the two guarded transitivity clauses of $(\gamma'_\emptyset)$. This however destroys the regularity property, and in fact no polynomial size regular refutations exist for GGT_n [1].

As usual, a *partial order* on $[n]$ is an antisymmetric, transitive relation binary relation on $[n]$. We shall be mostly interested in "partial specifications" of partial orders: partial specifications are not required to be transitive.

Definition 7. *A* partial specification*, τ, of a partial order is a set of ordered pairs $\tau \subseteq [n] \times [n]$ which are consistent with some (partial) order. The minimal partial order containing τ is the transitive closure of τ. We write $i \prec_\tau j$ to denote $\langle i, j\rangle \in \tau$, and write $i \prec_\tau^* j$ to denote that $\langle i, j\rangle$ is in the transitive closure of τ.*

The τ-minimal *elements are the i's such that $j \prec_\tau i$ does not hold for any j.*

We are primarily interested in particular kinds of partial orders, called "bipartite" partial orders, which do not have any chain of inequalities $x \prec y \prec z$.

Definition 8. *A* bipartite partial order *is a binary relation π on $[n]$ with disjoint domain and range. The set of π-minimal elements is denoted M_π.*

Figure 1 shows an example. The bipartiteness of π arises from the fact that M_π and $[n] \setminus M_\pi$ partition $[n]$ into two sets. Note that if $i \prec_\pi j$, then $i \in M_\pi$ and $j \notin M_\pi$. In addition, M_π contains the isolated points of π.

Definition 9. *Let τ be a specification of a partial order. The bipartite partial order π that is* associated with *τ is defined by letting $i \prec_\pi j$ hold for precisely those i and j such that i is τ-minimal and $i \prec_\tau^* j$.*

It is easy to check that π is a bipartite partial order. The intuition is that π retains only the information about whether $i \prec_\tau^* j$ for minimal elements i, and forgets the ordering that τ imposes on non-minimal elements. (See Fig. 1.)

Definition 10. *Let π be a bipartite partial order on $[n]$. Then* $\mathrm{GT}_{\pi,n}$ *is the set of clauses containing:*

(α) The clauses $\bigvee_{j\neq i} x_{j,i}$, for each value $i \in M_\pi$.
(β) The transitivity clauses $T_{i,j,k} := \overline{x}_{i,j} \vee \overline{x}_{j,k} \vee \overline{x}_{k,i}$ for all distinct i, j, k in M_π. (Vertices i, j, k' in Fig. 2 show an example.)
(γ) The transitivity clauses $T_{i,j,k}$ for all distinct i, j, k such that $i, j \in M_\pi$ and $i \not\prec_\pi k$ and $j \prec_\pi k$. (As shown in Fig. 2.)

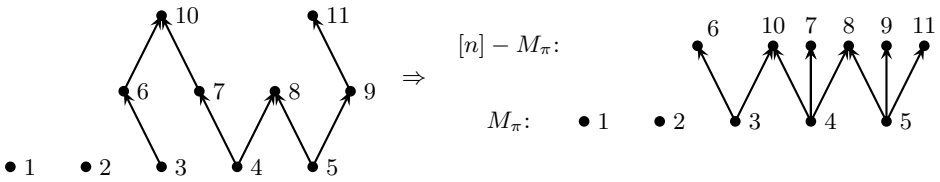


Fig. 1. Example of a partial specification of a partial order (left) and the associated bipartite partial order (right).

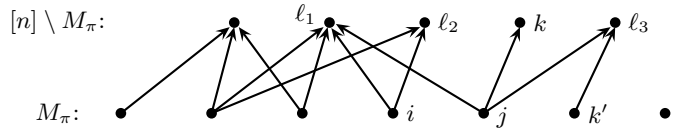


Fig. 2. A bipartite partial order π is pictured, with the ordered pairs of π shown as directed edges. (For instance, $j \prec_\pi k$ holds.) The nodes i, j, k shown are an example of nodes used for a transitivity axiom $\overline{x}_{i,j} \vee \overline{x}_{j,k} \vee \overline{x}_{k,i}$ of type (γ). The nodes i, j, k' are an example of the nodes for a transitivity axiom of type (β).

$\mathrm{GT}_{\pi,n}$ is satisfiable if π is nonempty, for example by the assignment that sets $x_{j,i}$ true for some fixed $j \notin M_\pi$ and every $i \in M_\pi$, and sets all other variables false. However, there is no assignment which satisfies $\mathrm{GT}_{\pi,n}$ and is consistent with π. This fact is proved by the regular derivation P_π of Lemma 1.

Definition 11. *For π a bipartite partial order, the clause $\left(\bigvee \overline{\pi}\right)$ is defined by*

$$\left(\bigvee \overline{\pi}\right) \;:=\; \{\overline{x}_{i,j} : i \prec_\pi j\},$$

Lemma 1. *Let π be a bipartite partial order on $[n]$. Then there is a regular derivation P_π of $\left(\bigvee \overline{\pi}\right)$ from the set $\mathrm{GT}_{\pi,n}$.*

The only variables resolved on in P_π are the following: the variables $x_{i,j}$ such that $i, j \in M_\pi$, and the variables $x_{i,k}$ such that $k \notin M_\pi$, $i \in M_\pi$, and $i \not\prec_\pi k$.

Lemma 1 implies that if π is the bipartite partial order associated with a partial specification τ of a partial order, then the derivation P_π does not resolve on any literal whose value is set by τ. This is proved by noting that if $i \prec_\tau j$, then $j \notin M_\pi$.

If π is empty, $M_\pi = [n]$ and there are no clauses of type (γ). In this case, $\mathrm{GT}_{\pi,n}$ is identical to GT_n, and P_π is the refutation of GT_n of Theorem 5.

Lemma 1 is proved similarly to Theorem 5, taking care to resolve on variables in the correct order. The proof is left to the full version of the paper.

Proof (of Theorem 1). We will construct a series of "LR partial refutations", denoted $R_0, R_1, R_2, \ldots$; this process eventually terminates with a pool refutation of GGT_n. The terminology "LR partial" indicates that the refutation is being constructed in left-to-right order, with the left part of the refutation properly

formed, but with many of the remaining leaves being labeled with bipartite partial orders instead of with valid learned clauses or initial clauses from GGT_n.

An LR partial refutation R is a tree with nodes labeled with clauses that form a correct pool resolution proof, except possibly at the leaves (the initial clauses). Furthermore, it must satisfy the following conditions.

a. R is a tree. The root is labeled with the empty clause. Each non-leaf node in R has a left child and right child; the clause labeling the node is derived by resolution from the clauses on its two children.
b. For C a clause occurring in R, define $\tau(C)$ to be the set of ordered pairs $\langle i, j\rangle$ such that $\overline{x}_{i,j} \in C^+$. Note that $C \subseteq C^+$ by definition. In many cases, $\tau(C)$ will be a partial specification of a partial order, but this is not always true. For instance, if C is a transitivity axiom, $\tau(C)$ has a 3-cycle and is not consistent as a specification of a partial order.
c. Leaves of R are flagged as "finished" or "unfinished".
d. Each finished leaf L is labeled with either a clause from GGT_n or a clause that occurs to the left of L in the postorder traversal of R.
e. For an unfinished leaf labeled with clause C, the set $\tau(C)$ is a partial specification of a partial order. Furthermore, letting π be the bipartite partial order associated with $\tau(C)$, the clause C is equal to $(\bigvee\overline{\pi})$.

Property e. is crucial for avoiding degenerate resolution inferences, and is a novel part of our construction. As shown below, each unfinished leaf, labeled with a clause $C = (\bigvee\overline{\pi})$, will be replaced by a derivation S. The derivation S often will be based on P_π, and thus might be expected to end with exactly the clause C; however, some of the resolution inferences needed for P_π might be disallowed by the pool property. So S will instead be a derivation of a clause C' such that $C \subseteq C' \subseteq C^+$. The condition $C' \subseteq C^+$ is required because any literal $x \in C' \setminus C$ will be handled by modifying the refutation R by propagating x downward in R until reaching a clause that already contains x. The condition $C' \subseteq C^+$ ensures that such a clause exists. The fact that $C' \supseteq C$ means that enough literals are present for the derivation to use only (non-degenerate) resolution inferences — indeed our constructions will pick C so that it contains the literals that must be present for use as resolution literals.

The construction begins by letting R_0 be the "empty" refutation, containing just the empty clause. Of course, this clause is an unfinished leaf, and $\tau(\emptyset) = \emptyset$.

Assume R_i has been already constructed, with C the leftmost unfinished clause. R_{i+1} will be formed by replacing C by a refutation S of some clause C' such that $C \subseteq C' \subseteq C^+$.

We need to describe the (LR partial) refutation S. By e., C is $(\bigvee\overline{\pi})$. The intuition is that we would like to let S be the derivation P_π of C from Lemma 1. The first difficulty with this is that P_π is dag-like, and the LR-refutation is intended to be tree-like. This difficulty, however, can be circumvented by just expanding P_π, which is regular, into a tree-like regular derivation with lemmas by the simple expedient of using a depth-first traversal of P_π. The second, and more serious, difficulty is that P_π is a derivation from GT_n, not GGT_n; namely, P_π

uses the transitivity clauses of GT_n instead of the guarded transitivity clauses of GGT_n. These transitivity clauses $T_{i,j,k}$ are handled one at a time treating them, as needed, with four separate cases. Case (*i*) requires no change to P_π; cases (*ii*) and (*iii*) require a small change; and case (*iv*) abandons the subproof P_π and instead "learns" the transitivity clause.

By the remark made after Lemma 1, no literal in C^+ is used as a resolution literal in P_π.

(*i*) If an initial transitivity clause of P_π already appears earlier in R_i (that is, to the left of C), then it is already *learned*, and can be used freely in P_π.

In the remaining cases (*ii*)-(*iv*), the transitivity clause $T_{i,j,k}$ is not yet learned. Let the guard variable for $T_{i,j,k}$ be $x_{r,s}$, so $r = r(i,j,k)$ and $s = s(i,j,k)$.

(*ii*) Suppose case (*i*) does not apply and that the guard variable $x_{r,s}$ or its negation $\overline{x}_{r,s}$ is a member of C^+. The guard variable thus is used as a resolution variable somewhere along the branch from the root to clause C. Then, as just argued above, Lemma 1 implies that $x_{r,s}$ is not resolved on in P_π. Therefore, we can add the literal $x_{r,s}$ or $\overline{x}_{r,s}$ (respectively) to the clause $T_{i,j,k}$ and to every clause on any path below $T_{i,j,k}$ until reaching a clause that already contains that literal. This replaces $T_{i,j,k}$ with one of the initial clauses $T_{i,j,k} \vee x_{r,s}$ or $T_{i,j,k} \vee \overline{x}_{r,s}$ of GGT_n. By construction, it preserves the validity of the resolution inferences of R_i as well as the regularity property. Note this adds the literal $x_{r,s}$ or $\overline{x}_{r,s}$ to the final clause C' of the modified P_π. This maintains the property that $C \subseteq C' \subseteq C^+$.

(*iii*) Suppose case (*i*) does not apply and that $x_{r,s}$ is not used as a resolution variable anywhere below $T_{i,j,k}$ in P_π and is not a member of C^+. In this case, P_π is modified so as to derive the clause $T_{i,j,k}$ from the two GGT_n clauses $T_{i,j,k} \vee x_{r,s}$ and $T_{i,j,k} \vee \overline{x}_{r,s}$ by resolving on $x_{r,s}$. This maintains the regularity of the derivation. And, henceforth $T_{i,j,k}$ will be learned.

If all of the transitivity clauses in P_π can be handled by cases (*i*)-(*iii*), then we use P_π to define R_{i+1}. Namely, let P'_π be the derivation P_π as modified by the applications of cases (*ii*) and (*iii*). The derivation P'_π is regular and dag-like, so we can recast it as a tree-like derivation S with lemmas, by using a depth-first traversal of P'_π. The size of S is linear in the size of P'_π, since the only new clauses in S are clauses which are repeated as lemmas and, as an overestimate, there are at most two lemmas per clause in P'_π. The final line of S is the clause C', namely C plus the literals introduced by case (*ii*). The derivation R_{i+1} is formed from R_i by replacing the clause C with the derivation S of C', and then propagating each new literal $x \in C' \setminus C$ downward, adding x to clauses below S until reaching a clause that already contains x. Since S contains no unfinished leaf, R_{i+1} contains one fewer unfinished leaves than R_i.

On the other hand, if even one transitivity axiom $T_{i,j,k}$ in P_π is not covered by the above three cases, then case (*iv*) must be used instead. This introduces a completely different construction to form S:

(*iv*) Let $T_{i,j,k}$ be any transitivity axiom in P_π that is not covered by cases (*i*)-(*iii*). The guard variable $x_{r,s}$ is used as a resolution variable in P_π somewhere below $T_{i,j,k}$; in general, this means we cannot use resolution on $x_{r,s}$ to derive $T_{i,j,k}$ while maintaining the pool property. Hence, P_π is no longer used, and we instead form S with a short left-branching path that "learns" $T_{i,j,k}$. This will generate two or three new unfinished leaf nodes. Since unfinished leaf nodes in a LR partial derivation must be labeled with clauses from bipartite partial orders, it is also necessary to attach short derivations to these unfinished leaf nodes to make the unfinished leaf clauses of S correspond correctly to bipartite partial orders. These unfinished leaf nodes are then kept in R_{i+1} to be handled at later stages. There are separate constructions depending on whether $T_{i,j,k}$ is a clause of type (β) or (γ); some of the details are given below.

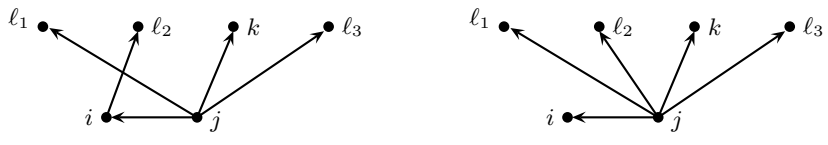
First suppose $T_{i,j,k}$ is of type (γ), and thus $\overline{x}_{j,k}$ appears in C. (Refer to Fig. 2.) Let $x_{r,s}$ be the guard variable for the transitivity axiom $T_{i,j,k}$. The derivation S will have the form

$$\dfrac{\dfrac{\dfrac{T_{i,j,k}, x_{r,s} \qquad T_{i,j,k}, \overline{x}_{r,s}}{T_{i,j,k}} \qquad \begin{matrix} S_1 \\ \vdots \\ \overline{x}_{i,j}, \overline{x}_{i,k}, \overline{\pi}_{\text{-}[jk;jR(i)]} \end{matrix}}{\overline{x}_{i,j}, \overline{x}_{j,k}, \overline{\pi}_{\text{-}[jk;jR(i)]}} \qquad \begin{matrix} S_2 \\ \vdots \\ \overline{x}_{j,i}, \overline{x}_{j,k}, \overline{\pi}_{\text{-}[jk;iR(j)]} \end{matrix}}{\overline{x}_{j,k}, \overline{\pi}_{\text{-}[jk]}}$$

The notation $\overline{\pi}_{\text{-}[jk]}$ denotes the disjunction of the negations of the literals in π omitting the literal $\overline{x}_{j,k}$. We write "$iR(j)$" to indicate literals $x_{i,\ell}$ such that $j \prec_\pi \ell$. (The "$R(j)$" means "range of j".) Thus $\overline{\pi}_{\text{-}[jk;iR(j)]}$ denotes the clause containing the negations of the literals in π, omitting $\overline{x}_{j,k}$ and any literals $\overline{x}_{i,\ell}$ such that $j \prec_\pi \ell$. The clause $\overline{\pi}_{\text{-}[jk;jR(i)]}$ is defined similarly, and the notation extends in the obvious way.

The upper leftmost inference of S is a resolution inference on the variable $x_{r,s}$. Since $T_{i,j,k}$ is not covered by either case (*i*) or (*ii*), the variable $x_{r,s}$ does not appear in or below clause C in R_i. Thus, this use of $x_{r,s}$ as a resolution variable does not violate regularity. Furthermore, since $T_{i,j,k}$ is of type (γ), we have $i \not\prec_{\tau(C)} j$, $j \not\prec_{\tau(C)} i$, $i \not\prec_{\tau(C)} k$, and $k \not\prec_{\tau(C)} i$. Thus the literals $x_{i,j}$ and $x_{i,k}$ do not appear in or below C, so they also can be resolved on without violating regularity.

Let C_1 and C_2 be the final clauses of S_1 and S_2, and let C_1^- be the clause below C_1 and above C. The set $\tau(C_2)$ is obtained by adding $\langle j, i\rangle$ to $\tau(C)$, and similarly $\tau(C_1^-)$ is $\tau(C)$ plus $\langle i, j\rangle$. Since $T_{i,j,k}$ is type (γ), we have $i, j \in M_\pi$. Therefore, since $\tau(C)$ is a partial specification of a partial order, $\tau(C_2)$ and $\tau(C_1^-)$ are also both partial specifications of partial orders. Let π_2 and π_1 be the bipartite orders associated with these two partial specifications (respectively). We will form the subproof S_1 so that it contains the clause $(\bigvee \overline{\pi}_1)$ as its only unfinished clause. This will require adding inferences in S_1 which add and remove the appropriate literals. The first step of this type already occurs in going up from C_1^- to C_1 since this has removed $\overline{x}_{j,k}$ and added $\overline{x}_{i,k}$, reflecting the fact that j is not π_1-minimal and thus $x_{i,k} \in \pi_1$ but $x_{j,k} \notin \pi_1$. Similarly, we will form S_2 so that its only unfinished clause is $(\bigvee \overline{\pi}_2)$.



(a) $\overline{x}_{j,k}, \overline{x}_{i,\ell_2}, \overline{x}_{j,i}, \overline{\pi}^*$ (b) $\overline{x}_{j,k}, \overline{x}_{i,\ell_2}, \overline{x}_{j,i}, \overline{\pi}^*$

Fig. 3. The partial orders for the fragment of S_2 shown in (1).

The situation for the subproof S_2 is shown in Fig. 3, which shows an extract from Fig. 2: the edges shown in part (a) of the figure correspond to the literals in the final line C_2 of S_2. Recall that literals $\overline{x}_{i,\ell}$ such that $j \prec_\pi \ell$ are omitted from the last line of S_2. (Correspondingly, the edge from i to ℓ_1 is omitted from Fig. 3.) C_2 may not correspond to a bipartite partial order as it may not partition $[n]$ into minimal and non-minimal elements; thus, C_2 may not qualify to be an unfinished node of R_{i+1}. (An example of this in Fig. 3(a) is that $j \prec_{\tau(C_2)} i \prec_{\tau(C_2)} \ell_2$, corresponding to $\overline{x}_{j,i}$ and $\overline{x}_{i,\ell_2}$ being in C_2.) The bipartite partial order π_2 associated with $\tau(C_2)$ is equal to the bipartite partial order that agrees with π except that each $i \prec_\pi \ell$ condition is replaced with the condition $j \prec_{\pi_2} \ell$. (This is represented in Fig. 3(b) by the fact that the edge from i to ℓ_2 has been replaced by the edge from j to ℓ_2. Note that the vertex i is no longer a minimal element of π_2; that is, $i \notin M_{\pi_2}$.) We wish to form S_2 to be a regular derivation of the clause $\overline{x}_{j,i}, \overline{\pi}_{\text{-}[jk;iR(j)]}$ from the clause $(\bigvee \overline{\pi}_2)$.

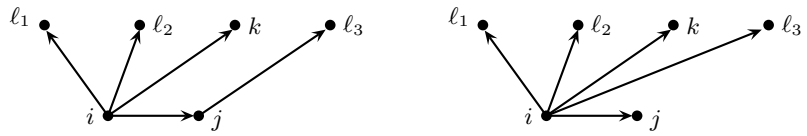
The subproof of S_2 for replacing $\overline{x}_{i,\ell_2}$ in $\overline{\pi}$ with $\overline{x}_{j,\ell_2}$ in $\overline{\pi}_2$ is

$$\dfrac{\begin{array}{cc} S_2' & \ddots \\ \overline{x}_{j,i}, \overline{x}_{i,\ell_2}, \overline{x}_{\ell_2,j} & \overline{x}_{j,k}, \overline{x}_{j,\ell_2}, \overline{x}_{j,i}, \overline{\pi}^* \end{array} \qquad \begin{array}{c} \ddots \text{ rest of } S_2 \\ \end{array}}{\overline{x}_{j,k}, \overline{x}_{i,\ell_2}, \overline{x}_{j,i}, \overline{\pi}^*} \tag{1}$$

where $\overline{\pi}^*$ is $\overline{\pi}_{\text{-}[jk;iR(j);i\ell_2]}$. The part labeled "rest of S_2" will handle similarly the other literals ℓ such that $i \prec_\pi \ell$ and $j \not\prec_\pi \ell$. The final line of S_2' is T_{j,i,ℓ_2}. This is a GT_n axiom, not a GGT_n axiom; however, it can be handled by the methods of cases (*i*)-(*iii*). Namely, if T_{j,i,ℓ_2} has already been learned by appearing somewhere to the left in R_i, then S_2' is just this single clause. Otherwise, let the guard variable for T_{j,i,ℓ_2} be $x_{r',s'}$. If $x_{r',s'}$ is used as a resolution variable below T_{j,i,ℓ_2}, then replace T_{j,i,ℓ_2} with $T_{j,i,\ell_2} \vee x_{r',s'}$ or $T_{j,i\ell_2} \vee \overline{x}_{r',s'}$, and propagate the $x_{r',s'}$ or $\overline{x}_{r',s'}$ to clauses down the branch leading to T_{j,i,ℓ_2} until reaching a clause that already contains that literal. Finally, if $x_{r',s'}$ has not been used as a resolution variable in R_i below C, then let S_2' consist of a resolution inference deriving (and learning) T_{j,i,ℓ_2} from the clauses $T_{j,i,\ell_2}, x_{r',s'}$ and $T_{j,i,\ell_2}, \overline{x}_{r',s'}$.

To complete the construction of S_2, the inference (1) is repeated for each value of ℓ such that $i \prec_\pi \ell$ and $j \not\prec_\pi \ell$. The result is that S_2 has one unfinished leaf clause, and it is labelled with the clause $(\bigvee \overline{\pi}_2)$.

We next describe the subproof S_1 of S. The situation is shown in Fig. 4. As in the formation of S_2, the final clause C_1 in S_1 may need to be modified in order to correspond to the bipartite partial order π_1 which is associated with $\tau(C_1)$. First, note that the literal $\overline{x}_{j,k}$ is already replaced by $\overline{x}_{i,k}$ in the final clause



(a) $\overline{x}_{i,k}, \overline{x}_{j,\ell_3}, \overline{x}_{i,j}, \overline{\pi}^*$ (b) $\overline{x}_{i,k}, \overline{x}_{i,\ell_3}, \overline{x}_{i,j}, \overline{\pi}^*$

Fig. 4. The partial orders for the fragment of S_1 shown in (2).

of S_1. The other change that is needed is that, for every ℓ such that $j \prec_\pi \ell$ and $i \not\prec_\pi \ell$, we must replace $\overline{x}_{j,\ell}$ with $\overline{x}_{i,\ell}$ since we have $j \not\prec_{\pi_1} \ell$ and $i \prec_{\pi_1} \ell$. Vertex ℓ_3 in Fig. 4 is an example of a such a value ℓ. The ordering in the final clause of S_1 is shown in part (a), and the desired ordered pairs of π_1 are shown in part (b). Note that j is no longer a minimal element in π_1.

The replacement of $\overline{x}_{j,\ell_3}$ with $\overline{x}_{i,\ell_3}$ is effected by the following inference, letting $\overline{\pi}^*$ now be $\overline{\pi}_{\text{-}[jk;jR(i);j\ell_3]}$.

$$\dfrac{\begin{array}{cc} S_1' \cdot \vdots \cdot & \cdot \vdots \cdot \text{ rest of } S_1 \\ \overline{x}_{i,j}, \overline{x}_{j,\ell_3}, \overline{x}_{\ell_3,i} & \overline{x}_{i,k}, \overline{x}_{i,\ell_3}, \overline{x}_{i,j}, \overline{\pi}^* \end{array}}{\overline{x}_{i,k}, \overline{x}_{j,\ell_3}, \overline{x}_{i,j}, \overline{\pi}^*} \tag{2}$$

The "rest of S_1" will handle similarly the other literals ℓ such that $j \prec_\pi \ell$ and $i \not\prec_\pi \ell$. Note that the final clause of S_1' is the transitivity axiom T_{i,j,ℓ_3}. The subproof S_1' is formed in the same way that S_2' was formed above. Namely, depending on the status of the guard variable $x_{r',s'}$ for T_{i,j,ℓ_3}, one of the following is done: (*i*) the clause T_{i,j,ℓ_3} is already learned and can be used as is, or (*ii*) one of $x_{r',s'}$ or $\overline{x}_{r',s'}$ is added to the clause and propagated down the proof, or (*iii*) the clause T_{i,j,ℓ_3} is inferred using resolution on $x_{r',s'}$ and becomes learned.

To complete the construction of S_1, the inference (2) is repeated for each value of ℓ such that $j \prec_\pi \ell$ and $i \not\prec_\pi \ell$. The result is that S_1 has one unfinished leaf clause, and it corresponds to the bipartite partial order π_1.

That completes the construction of the subproof S for the subcase of (*iv*) where $T_{i,j,k}$ is of type (γ). Now suppose $T_{i,j,k}$ is of type (β). (For instance, the values i, j, k' of Fig. 2.) In this case the derivation S will have the form

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{T_{i,j,k}, x_{r,s} \quad T_{i,j,k}, \overline{x}_{r,s}}{T_{i,j,k}} \quad \begin{array}{c} S_3 \cdot \vdots \cdot \\ \overline{x}_{i,j}, \overline{x}_{i,k}, \overline{\pi}_{\text{-}[jR(i),kR(i\cup j)]} \end{array}}{\overline{x}_{i,j}, \overline{x}_{j,k}, \overline{\pi}_{\text{-}[jR(i),kR(i\cup j)]}} \quad \begin{array}{c} S_4 \cdot \vdots \cdot \\ \overline{x}_{i,j}, \overline{x}_{k,j}, \overline{\pi}_{\text{-}[jR(i\cap k)]} \end{array}}{\overline{x}_{i,j}, \overline{\pi}_{\text{-}[jR(i\cap k)]}} \quad \begin{array}{c} S_5 \cdot \vdots \cdot \\ \overline{x}_{j,i}, \overline{\pi}_{\text{-}[iR(j)]} \end{array}}{\overline{\pi}}$$

where $x_{r,s}$ is the guard variable for $T_{i,j,k}$. We write $[\overline{\pi}_{\text{-}[jR(i\cap k)]}]$ to mean the negations of literals in π omitting any literal $\overline{x}_{j,\ell}$ such that $i \prec_\pi \ell$ and $k \prec_\pi \ell$. Similarly, $\overline{\pi}_{\text{-}[jR(i),kR(i\cup j)]}$ indicates the negations of literals in π, omitting the literals $\overline{x}_{j,\ell}$ such that $i \prec_\pi \ell$ and the literals $\overline{x}_{k,\ell}$ such that $i \prec_\pi \ell$ or $j \prec_\pi \ell$.

Note that the resolution on $x_{r,s}$ used to derive $T_{i,j,k}$ does not violate regularity, since otherwise $T_{i,j,k}$ would have been covered by case (*ii*). Likewise, the resolutions on $x_{i,j}$, $x_{i,k}$, and $x_{j,k}$ do not violate regularity since $T_{i,j,k}$ is of type (β).

The subproofs S_3, S_4, and S_5 are handled similarly to the way the subproofs S_1 and S_2 were handled above, albeit with some extra complications in the S_4 case. The detailed constructions are in the full version of the paper.

Once some R_i has no unfinished clauses, we have the desired pool refutation. We claim that the process stops after polynomially many stages.

To prove this, recall that R_{i+1} is formed by handling the leftmost unfinished clause using one of cases (*i*)-(*iv*). In the first three cases, the unfinished clause is replaced by a derivation based on P_π. Since P_π has size $O(n^3)$, this means that the number of clauses in R_{i+1} is at most the number of clauses in R_i plus $O(n^3)$. Also, by construction, R_{i+1} has one fewer unfinished clauses than R_i. In case (*iv*) however, R_{i+1} is formed by adding up to $O(n)$ many clauses to R_i plus adding either two or three new unfinished leaf clauses. However, case (*iv*) always causes at least one transitivity axiom $T_{i,j,k}$ to be learned. Therefore, case (*iv*) can occur at most $2\binom{n}{3} = O(n^3)$ times. Consequently at most $3 \cdot 2\binom{n}{3} = O(n^3)$ many unfinished clauses are added throughout the entire process. It follows that the process stops with R_i having no unfinished clauses for some $i \leq 6\binom{n}{3} = O(n^3)$. Therefore there is a pool refutation of GGT_n with $O(n^6)$ lines.

By inspection, each clause in the refutation contains $O(n^2)$ literals. This is because the largest clauses are those corresponding to (small modifications of) bipartite partial orders, and because bipartite partial orders can contain at most $O(n^2)$ many ordered pairs. Furthermore, the refutations P_n for the graph tautology GT_n contain only clauses of size $O(n^2)$. Q.E.D. Theorem 1

The proofs of Theorems 2 and 3 are left to the full version of the paper, but use similar methods. Theorem 4 follows from the algorithm implicit in the proof of Theorem 3. The following gives a sketch of the algorithm for DPLL search with clause learning which always succeeds in finding a refutation of the GGT_n clauses. At each point in the DPLL search procedure, there is a partial assignment τ, and the search algorithm must do one of the following:

(1) If unit propagation yields a contradiction, then learn a clause $T_{i,j,k}$ if possible, and backtrack.
(2) Otherwise, if there are any literals in the bipartite partial order π associated with τ which are not assigned a value, branch on one of these literals to set its value.
(3) Otherwise, determine whether there is a clause $T_{i,j,k}$ which is used in the proof P_π whose guard literals are resolved on in P_π. (See Lemma 1.) If not, do a DPLL traversal of P_π, eventually backtracking from the assignment τ.
(4) Otherwise, let $T_{i,j,k}$ block P_π from being traversed, and branch on its variables in the order given in the above proof. From this, learn the clause $T_{i,j,k}$.

Acknowledgements. We thank J. Hoffmann and J. Johannsen for a correction to an earlier version of the proof of Theorem 2, and A. Van Gelder, A. Beckmann, and T. Pitassi for encouragement, suggestions, and comments.

References

1. Alekhnovich, M., Johannsen, J., Pitassi, T., Urquhart, A.: An exponential separation between regular and general resolution. Theory of Computation 3(4), 81–102 (2007)
2. Atserias, A., Fichte, J.K., Thurley, M.: Clause-learning algorithms with many restarts and and bounded-width resolution. Journal of Artificial Intelligence Research 40, 353–373 (2011)
3. Beame, P., Kautz, H.A., Sabharwal, A.: Towards understanding and harnessing the potential of clause learning. J. Artificial Intelligence Research 22, 319–351 (2004)
4. Beckmann, A., Buss, S.R.: Separation results for the size of constant-depth propositional proofs. Annals of Pure and Applied Logic 136, 30–55 (2005)
5. Bonet, M.L., Galesi, N.: A study of proof search algorithms for resolution and polynomial calculus. In: 40th Annual IEEE Symp. on Foundations of Computer Science. pp. 422–431. IEEE Computer Society (1999)
6. Buss, S.R.: Pool resolution is NP-hard to recognise. Archive for Mathematical Logic 48(8), 793–798 (2009)
7. Buss, S.R., Hoffmann, J., Johannsen, J.: Resolution trees with lemmas: Resolution refinements that characterize DLL-algorithms with clause learning. Logical Methods of Computer Science 4, 4:13(4:13), 1–18 (2008)
8. Goerdt, A.: Regular resolution versus unrestricted resolution. SIAM Journal on Computing 22(4), 661–683 (1993)
9. Hertel, P., Bacchus, F., Pitassi, T., Van Gelder, A.: Clause learning can effectively p-simulate general propositional resolution. In: Proc. 23rd AAAI Conf. on Artificial Intelligence (AAAI 2008). pp. 283–290. AAAI Press (2008)
10. Huang, W., Yu, X.: A DNF without regular shortest consensus path. SIAM Journal on Computing 16(5), 836–840 (1987)
11. Krishnamurthy, B.: Short proofs for tricky formulas. Acta Informatica 22(3), 253–275 (1985)
12. Pipatsrisawat, K., Darwiche, A.: On the power of clause-learning sat solvers as resolution engines. Artificial Intelligence 172(2), 512–525 (2011)
13. Segerlind, N., Buss, S.R., Impagliazzo, R.: A switching lemma for small restrictions and lower bounds for k-DNF resolution. SIAM Journal on Computing 33(5), 1171–1200 (2004)
14. Stålmarck, G.: Short resolution proofs for a sequence of tricky formulas. Acta Informatica 33(3), 277–280 (1996)
15. Urquhart, A.: A near-optimal separation of regular and general resolution. SIAM Journal on Computing 40(1), 107–121 (2011)
16. Van Gelder, A.: Pool resolution and its relation to regular resolution and DPLL with clause learning. In: Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2005). pp. 580–594. Lecture Notes in Computer Science 3835, Springer-Verlag (2005)
17. Van Gelder, A.: Preliminary report on input cover number as a metric for propositional resolution proofs. In: Theory and Applications of Satisfiability Testing - SAT 2006. pp. 48–53. Lecture Notes in Computer Science 4121, Springer Verlag (2006)