Journal of Artificial Intelligence Research 49 (2014) 669-703 Submitted 11/13; published 4/14

Improved Separations of Regular Resolution from Clause Learning Proof Systems

Maria Luisa Bonet BONET@LSI.UPC.EDU
Lenguajes y Sistemas Informáticos,
Universidad Politécnica de Cataluña,
Barcelona, Spain

Sam Buss SBUSS@MATH.UCSD.EDU
Department of Mathematics,
University of California, San Diego,
La Jolla, CA 92093-0112, USA

Jan Johannsen JAN.JOHANNSEN@IFI.LMU.DE
Institut für Informatik,
Ludwig-Maximilians Universität München,
D-80538 München, Germany

Abstract

This paper studies the relationship between resolution and conflict driven clause learning (CDCL) without restarts, and refutes some conjectured possible separations. We prove that the guarded, xor-ified pebbling tautology clauses, which Urquhart proved are hard for regular resolution, as well as the guarded graph tautology clauses of Alekhnovich, Johannsen, Pitassi, and Urquhart have polynomial size pool resolution refutations that use only input lemmas as learned clauses. For the latter set of clauses, we extend this to prove that a CDCL search without restarts can refute these clauses in polynomial time, provided it makes the right choices for decision literals and clause learning. This holds even if the CDCL search is required to greedily process conflicts arising from unit propagation. This refutes the conjecture that the guarded graph tautology clauses or the guarded xor-ified pebbling tautology clauses can be used to separate CDCL without restarts from general resolution. Together with subsequent results by Buss and Kołodziejczyk, this means we lack any good conjectures about how to establish the exact logical strength of conflict-driven clause learning without restarts.

1. Introduction

The problem SAT of deciding the satisfiability of propositional CNF formulas is of great theoretical and practical interest. Even though SAT is NP-complete, industrial instances with hundreds of thousands variables are routinely solved by state-of-the-art SAT solvers. Most of these solvers use conflict-driven clause learning (CDCL) based on the work of Marques-Silva and Sakallah (1999). These CDCL solvers use the DPLL (Davis-Putnam-Logemann-Loveland) search procedure with clause learning, extended with additional techniques such as fast backtracking, restarts, and variable selection heuristics.

Without clause learning, the DPLL procedure is equivalent to tree-like resolution. With the addition of clause learning,[1] CDCL becomes considerably more powerful. In fact, CDCL together with unlimited restarts is capable of polynomially simulating general resolution proofs (Pipatsrisawat & Darwiche, 2011). Without restarts, CDCL is known to polynomially simulate regular resolution (Buss, Hoffmann, & Johannsen, 2008). Furthermore, general resolution is known to be strictly stronger than regular resolution (Alekhnovich, Johannsen, Pitassi, & Urquhart, 2007). However, the exact power of CDCL without restarts is unknown. This question is interesting not just because CDCL without restarts is a core search method for most SAT aolvers, but also because a better understanding of the power of CDCL may lead to a better understanding of the practical performance of SAT solvers.

Alekhnovich et al. (2007) and Urquhart (2011) gave three examples of unsatisfiable sets of clauses that require exponentially longer regular resolution refutations than (general) resolution refutations. In view of the fact that CDCL without restarts lies between regular resolution and resolution, these three examples were conjecturally good candidates for showing that CDCL without restarts cannot polynomially simulate general resolution. The present paper refutes these conjectures for two of these examples; namely, we prove that CDCL without restarts can give polynomial size refutations of the guarded graph tautologies clauses of Alekhnovich et al. and the guarded xor-ified pebbling tautologies clauses of Urquhart, provided the CDCL search makes optimal choices for decision literals and for learning clauses and forgetting learned clauses. For the former tautology, we further show that the CDCL search can be required to be greedy and never ignore contradictions that can be found by unit propagation. It follows that those two tautologies do not give a superpolynomial separation of resolution refutations from CDCL refutations without restarts. Buss and Kołodziejczyk (2012) subsequently proved a similar result for the Stone tautologies, which Alekhnovich et al. proved give an exponential separation of regular resolution and resolution. Thus, there are presently no conjectured examples of tautologies that would provide an exponential separation between the power of CDCL without restarts and the power of resolution. On the other hand, it looks very difficult to prove that CDCL without restarts can polynomially simulate resolution. We consequently lack good conjectures about how to characterize the exact strength of CDCL without restarts.

Beame, Kautz, and Sabharwal (2004) gave the first theoretical analysis of CDCL. Among other things, they noted that CDCL with restarts simulates general resolution. Their construction, however, was rather unnatural as it requires the CDCL algorithm to ignore some contradictions. This situation was rectified by Pipatsrisawat and Darwiche (2011), who showed that CDCL solvers with restarts which use unit propagation and never ignore contradictions can also simulate resolution. Their proof was based on the technique of absorption that was first defined by Atserias, Fichte, and Thurley (2011).

Beame et al. (2004) also studied CDCL without restarts. Using "proof trace extensions", they showed that CDCL without restarts is strictly stronger than any "natural" proof system strictly weaker than resolution. A *natural* proof system is one in which proofs do not increase in length superpolynomially when variables are restricted to constants; natural

1. In this paper, we use "CDCL" as a synonym for "DPLL with clause learning". When we discuss whether CDCL without restarts can polynomially simulate resolution, we mean whether the simulation is possible with the correct choices for decision literals and learned clauses.

proof systems include systems such as tree-like and regular resolution. The proof trace method changes the formulas by introducing extraneous variables and clauses, which have the effect of giving CDCL more freedom in choosing decision variables for branching.

Buss et al. (2008) and Hertel, Bacchus, Pitassi, and Van Gelder (2008) gave improved versions of the proof trace extension method so that the extraneous variables depend only on the set of clauses being refuted and not on the resolution refutation of the clauses. The drawback remains, however, that the proof trace extension method gives contrived sets of clauses and contrived resolution refutations, and consequently does not give much insight into the power of CDCL.

There have been two approaches to formalizing CDCL without restarts as a static proof system rather than as a proof search algorithm. The first is pool resolution with a degenerate resolution inference, due to Van Gelder (2005) and studied further by Hertel et al. (2008). 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 one or both of the hypotheses may be lacking occurrences of the resolution literal. (Detailed definitions are given in Section 2.) Van Gelder argued that pool resolution with degenerate resolution inferences simulates a wide range of CDCL algorithms without restarts. He also gave a proof, using techniques of Alekhnovich et al. (2007), that pool resolution with degenerate inferences is stronger than regular resolution, using extraneous variables similar to proof trace extensions.

The second approach is due to Buss et al. (2008) who introduced a different degenerate resolution rule called w-resolution, and a proof system, called regWRTI, based on w-resolution and clause learning of "input lemmas". They proved that regWRTI exactly captures non-greedy CDCL without restarts. As discussed below, "non-greedy" means that contradictions may need to be ignored by the CDCL search.

It remains open whether any of CDCL without restarts, pool resolution (with or without degenerate inferences), or the regWRTI proof system can polynomially simulate general resolution. One approach to answering these questions is to try to separate pool resolution or regWRTI from general resolution. However, the best so-far obtained separations from resolution apply only to the weaker system of regular resolution, based on work of Alekhnovich et al. (2007) and Urquhart (2011) giving exponential separations between regular resolution and general resolution. Alekhnovich et al. proved their exponential separation of regular resolution and resolution for two families of tautologies, variants of the graph tautology clauses GT' and the "Stone" pebbling tautology clauses. Urquhart subsequently gave a related separation using a different set of pebbling tautology clauses which he denoted denoted Π_i.[2] The present paper calls the GT' clauses the *guarded graph tautology clauses*, and denotes them GGT instead of GT'; their definition is given in Section 4. Section 3 defines the *guarded xor-ified pebbling tautology clauses* $\mathrm{GPeb}^{k\oplus}(G)$, which are essentially the same as the clauses Π_i.

An obvious question is whether pool resolution or regWRTI has polynomial size refutations of the GGT, $\mathrm{GPeb}^{k\oplus}$, or Stone clauses. The present paper resolves the first two questions by showing that both pool resolution and regRTI do indeed have polynomial size

2. Huang and Yu (1987) also gave a separation of regular resolution and general resolution, but only for a single set of clauses. Goerdt (1993) gave a quasipolynomial separation of regular resolution and general resolution.

refutations of the GGT and $\mathrm{GPeb}^{k\oplus}$ clauses. The refutations avoid the use of extraneous variables in the style of proof trace extensions; furthermore, they use only the traditional resolution rule and do not require degenerate resolution inferences or w-resolution inferences. In addition, we use only learning of input clauses; thus, our refutations are also regWRTI refutations (and in fact regRTI refutations) in the terminology of Buss et al. (2008). As a corollary of the characterization of regWRTI by Buss et al., GGT and $\mathrm{GPeb}^{k\oplus}$ have polynomial size refutations that can be found by CDCL without restarts.

The Stone clauses have recently been shown to also have regRTI refutations by Buss and Kołodziejczyk (2012); although they use a rather different method than the present paper. Thus, none of the three principles separate CDCL without restarts from general resolution. It is natural to speculate that perhaps pool resolution, regWRTI, or CDCL without restarts can simulate general resolution. However, it is hard to be optimistic that such simulations exist, as we have been unable to extend our methods or those of Buss and Kołodziejczyk to give a polynomial simulation of general resolution by CDCL without restarts.

Our results are proved by giving regRTI refutations and then invoking a result of Buss et al. (2008, Thm. 5.6) which states that a regWRTI refutation of size n of a set Γ of clauses can be translated into a CDCL refutation that does not use restarts and has runtime polynomially bounded by n. The mentioned theorem of Buss et al. applies to CDCL algorithms that can learn clauses using the framework of Marques-Silva and Sakallah (1999) (see also Beame et al., 2004), but does make some important assumptions. The first assumption is that the CDCL algorithm makes optimal choices for decision literals and for learned clauses; the second assumption is that the CDCL algorithm may be non-greedy.

Informally, the fact that the CDCL search must make optimal choices for decision literals and learned clauses means that Buss et al. (2008) prove the equivalence of regWRTI with *nondeterministic* CDCL search without restarts. This assumption of nondeterminism is probably unavoidable in light of the conditional non-automatizability of resolution proved by Alekhnovich and Razborov (2001). However, it is a very reasonable assumption for characterizing CDCL search in terms of a formal proof system. Furthermore, lower bounds on the sizes of regWRTI refutations will imply lower bounds on the runtimes of CDCL without restarts.

A CDCL search is called *non-greedy* if it is allowed to ignore contradictions while continuing to assign further decision literals. Implemented CDCL algorithms are always greedy; namely, they learn conflicts and backtrack whenever possible; and it is probably a rare event that non-greedy CDCL algorithms would consistently outperform greedy algorithms, at least in practical applications. (However, this is an open question.) The upper bounds for CDCL without restarts obtained via upper bounds on regWRTI refutations potentially give only non-greedy CDCL searches. For the guarded graph tautologies, however, we prove more, namely that greedy and unit propagating CDCL without restarts can give polynomial size refutations of the guarded graph tautology clauses provided it makes optimal choices for decision literals and for learning and forgetting clauses.

We conjecture that the guarded xor-ified pebbling tautologies $\mathrm{GPeb}^{k\oplus}$ can also be refuted by polynomial size greedy and unit propagating CDCL without restarts. Preliminary investigations reveal no obstacle; however, the technical details are quite involved, and due to the length of the present paper, we have not carried out the complete construction. Each tautology seems to require a separate proof. Indeed, it is open whether greedy and unit propagating CDCL without restarts can simulate arbitrary regRTI proofs, or even arbitrary (dag-like) regular resolution refutations.

The outline of the paper is as follows. Section 2 defines resolution, degenerate resolution, and w-resolution, and then regular, tree, and pool resolution. It concludes with the definition of greedy and unit propagating. Section 3 defines the GPeb tautologies, including "xor-ification" and "guarded" initial clauses. It then proves the existence of polynomial size pool resolution and regRTI refutations of the guarded xor-ified $\mathrm{GPeb}^{k\oplus}$ clauses. The first idea of the proof is to try to follow the regular refutations of the unguarded $\mathrm{Peb}^{k\oplus}$ clauses. These refutations cannot be used directly however, since the initial clauses of $\mathrm{Peb}^{k\oplus}$ are "guarded" in the $\mathrm{GPeb}^{k\oplus}$ clauses and this yields refutations which violate the regularity/pool property. So, the second idea is that the proof search branches as needed to learn the initial unguarded $\mathrm{Peb}^{k\oplus}$ clauses. This generates additional clauses that must be proved, and the tricky part is to be sure that exactly the right set of additional clauses is generated.

Section 4 turns to the graph tautology clauses GT_n and their guarded versions, GGT_n. It first defines these clauses and states the main theorems about polynomial size refutations of the GGT_n clauses in pool resolution and regRTI. Section 4.1 defines the notion of bipartite partial order, and discusses the regular refutations of the graph tautology clauses GT_n as given by Stålmarck (1996) and Bonet and Galesi (2001). Section 4.2 constructs the pool/regRTI refutations of the GGT_n clauses. The intuition for this construction is similar to the constructions for the pebbling tautologies, but the technical details are much more involved. Section 5 concludes with an explicit description of a polynomial time greedy and unit propagating CDCL search without restarts which refutes the GGT_n clauses.

This paper is a reworking and an expansion of an extended abstract (Bonet & Buss, 2012a) and an unpublished preprint (Bonet & Buss, 2012b) by the first two authors. These earlier versions included only the results for the GGT tautologies and did not consider the GPeb principles.

2. Preliminaries

Propositional formulas are defined over a set of variables and the connectives $\wedge$, $\vee$ and $\neg$. We use the notation $\overline{x}$ to express the negation $\neg x$ 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 of its members. The empty clause, $\Box$, has truth value *False*. We shall only use formulas in *conjunctive normal form*, CNF; namely, a formula will be a set (conjunction) of clauses. We often use disjunction ($\vee$), union ($\cup$), and comma (,) interchangeably.

Definition 1. The three forms of resolution defined below take two clauses A and B called the *premises* and a literal x called the *resolution variable*, and produce a new clause C called the *resolvent*.

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

In all cases, it is required that $\overline{x} \notin A$ and $x \notin B$. The different forms of resolution are:

Resolution rule. The hypotheses have the forms $A := A' \vee x$ and $B := B' \vee \overline{x}$. The resolvent C is $A' \vee B'$.

Degenerate resolution rule. (Van Gelder, 2005; Hertel et al., 2008) 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. (Buss et al., 2008) From A and B as above, we infer the clause $C := (A \setminus \{x\}) \vee (B \setminus \{\overline{x}\})$. If the literal $x \notin A$ (resp., $\overline{x} \notin B$), then it is called a *phantom literal* of A (resp., B).

Degenerate and w-resolution combine weakening with resolution. Of course, it is well-known that adding weakening to resolution does not increase the refutational strength of resolution; however, the point of allowing degenerate or w-resolution is that the derivation may "learn" some clauses in the parts of the derivation that would have otherwise been pruned away if weakening were not allowed. For this reason, degenerate and w-resolution actually correspond better to DPLL search than resolution does.

Definition 2. A *resolution derivation*, or *proof*, of a clause C from a CNF formula F is a sequence of clauses $C_1, \dots, C_s$ such that $C = C_s$ and such that each clause from the sequence is either a clause from F or is the resolution resolvent of two previous clauses. If the derived clause, C_s, is the empty clause, this is called a *resolution refutation* of F. The more general concepts of degenerate and w-resolution derivations and refutations are defined similarly. The *size* of a proof is the number of clauses in the proof.

We use the terms "proof" and "derivation" interchangeably. A derivation is represented as a directed acyclic graph (dag) on the vertices $C_1, \dots, C_s$, where each clause from F has out-degree 0, and all the other vertices from $C_1, \dots, C_s$ have edges pointing to the two clauses from which they were derived. The empty clause has in-degree 0.

Resolution is sound and complete in the refutational sense: a CNF formula F has a refutation if and only if F is unsatisfiable. Furthermore, if there is a derivation of a clause C from F, then C is a consequence of F; that is, for every truth assignment σ, if σ satisfies F then it satisfies C. Conversely, if C is a consequence of F then there is a derivation of some $C' \subseteq C$ from F.

A resolution refutation is *regular* provided that, along any path in the directed acyclic graph, each variable is resolved on at most once. A resolution derivation of a clause C is *regular* provided that, in addition, no variable appearing in C is used as a resolution variable in the derivation. A refutation is *tree-like* if the underlying graph is a tree, so each occurrence of a clause in the refutation is used at most once as a premise of an inference.

We next define a version of pool resolution, using the conventions of Buss et al. (2008) who called this "tree-like regular resolution with lemmas" or "regRTL". The idea is that clauses obtained previously in the proof can be used freely as learned lemmas. To be able to talk about clauses previously obtained, we need to define an ordering of clauses.

Definition 3. Given a tree T, the *post-order* ordering $<_T$ of the nodes is defined as follows: if u is a node of T, v is a node in the subtree rooted at the left child of u, and w is a node in the subtree rooted at the right child of u, then $v <_T w <_T u$.

Definition 4. A *pool resolution* proof (also called a regRTL proof) from a set of initial clauses F is a resolution proof tree T that fulfills the following conditions: (a) each leaf is labeled with either a clause of F or 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. If the labeling of the root is the empty clause $\Box$, the pool resolution proof is a *pool refutation*.

The notions of *degenerate pool resolution* proof and *pool w-resolution* proof are defined similarly, but allowing degenerate resolution or w-resolution inferences, respectively. Van Gelder (2005) and Hertel et al. (2008) 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 by Buss (2009), however. It is also equivalent to the system regRTL defined by Buss et al. (2008). Pool w-resolution is the same as the system regWRTL of Buss et al. It is open whether the systems of pool resolution, degenerate pool resolution, and pool w-resolution are distinct. The present paper give examples of superpolynomial separations between these three systems and regular resolution.

A "lemma" in clause (a) of the above definition 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. Input resolution is the same as the "trivial resolution" of Beame et al. (2004), who used it to characterize clauses that can be learned from conflicts found by unit propagation (see also Chang, 1970). The use of input subderivations for learning clauses in pool resolution proofs is due to Buss et al. (2008). In their terminology, a pool resolution proof which uses only input lemmas is called a regRTI proof. Likewise a regWRTL proof that uses only input lemmas is called a regWRTI proof. To understand the nomenclature; "reg" stands for "regular", "W" for "w-resolution", "RT" for "resolution tree", "L" for lemma, and "I" for "input lemma".

Based on the definition of Van Gelder (2005), C^{pool} is defined as the "pool" of falsified literals at clause C. The definition of the "pool" includes the phantom literals used in w-resolution inferences:

Definition 5. Let R be a tree-like, regular refutation with lemmas using degenerate resolution, w-resolution, or resolution. Let C be a clause in R. Then, the clause C^{pool} is defined to equal

$$\begin{aligned} C^{\text{pool}} \quad := \quad \{x : & \text{ the literal } x \text{ occurs, either explicitly or as a phantom literal,} \\ & \text{in some clause of } R \text{ that lies on the branch} \\ & \text{from the root node of } R \text{ up to and including } C\}, \end{aligned}$$

Note that $C \subseteq C^{\text{pool}}$, and the regularity of R ensures that C^{pool} contains no contradictory literals.

3. Guarded, Xor-ified, Pebbling Principles

This section gives polynomial size regRTI refutations for the guarded pebbling tautology clauses which Urquhart (2011) proved require exponential size regular resolution proofs.

Definition 6. A *pointed dag* $G = (V, E)$ is a directed acyclic graph with a single sink t such that every vertex in G has indegree either 0 or 2. The pebbling tautology clauses $\mathrm{Peb}(G)$ for a pointed dag G are the following unsatisfiable set of clauses in the variables x_v for $v \in V$:

(α) x_s, for every source $s \in V$,

(β) $\overline{x}_u \vee \overline{x}_v \vee x_w$, for every vertex w with two (immediate) predecessors u and v,

(γ) $\overline{x}_t$, for t the sink vertex.

The clauses $\mathrm{Peb}(G)$ are Horn clauses and hence have a short tree-like resolution refutation of linear size. However, these clauses can be made difficult to refute by using "or-ification" or "xor-ification" (see Ben-Sasson, Impagliazzo, & Wigderson, 2004; Ben-Sasson, 2009, and Urquhart, 2011). Here we define Urquhart's (2011) "xor-ification" of a pebbling tautology clause. Xor-ification, for two variables, is due to Alekhnovich and Razborov as discussed by Ben-Sasson (2009), and is similar to the "or-ification" used by Ben-Sasson et al. (2004) which replaces each variable by the disjunction of two variables. The intuition for xor-ification is that each variable x_u is replaced by a set of clauses which expresses the exclusive or $x_{u,1} \oplus \cdots \oplus x_{u,k}$ of k new variables.

Definition 7. Let $k > 0$, and x_u be a variable of $\mathrm{Peb}(G)$. Let $x_{u,1}, \ldots, x_{u,k}$ be new variables, and let $x^1_{u,j}$ be $x_{u,j}$, and $x^{-1}_{u,j}$ be its complement $\overline{x}_{u,j}$. Define $x^{k\oplus}_u$ to be the set of clauses of the form

$$x^{i_1}_{u,1} \vee x^{i_2}_{u,2} \vee \cdots \vee x^{i_k}_{u,k} \tag{1}$$

where an even number of the values i_j equal -1 (and the rest equal 1). Dually, define $\overline{x}^{k\oplus}_u$ to be the set of clauses of the form (1) with an odd number of the i_j's equal to -1. Note there are 2^{k-1} clauses in each of $x^{k\oplus}_u$ and $\overline{x}^{k\oplus}_u$. If C is a clause $C = z_1 \vee \cdots \vee z_\ell$, each z_i a literal x_u or $\overline{x}_u$, then $C^{k\oplus}$ is the set of clauses of the form

$$C_1 \vee C_2 \vee \cdots \vee C_\ell,$$

where each $C_i \in z^{k\oplus}_i$. There are $2^{(k-1)\ell}$ many clauses in $C^{k\oplus}$.

Definition 8. The *xor-ified pebbling tautology clauses* $\mathrm{Peb}^{k\oplus}(G)$ is the set of clauses $C^{k\oplus}$ for $C \in \mathrm{Peb}(G)$. If G has n vertices, $\mathrm{Peb}^{k\oplus}(G)$ has $O(2^{3k}n)$ clauses.

Definition 9. Let G be a pointed graph with n vertices and $k = k(n) > 0$. Let ρ be a function with domain the set of clauses of $\mathrm{Peb}^{k\oplus}(G)$ and range the set of variables $x_{u,i}$ of $\mathrm{Peb}^{k\oplus}(G)$, such that, for all C, the variable $\rho(C)$ is not used in C. The *guarded xor-ified pebbling tautology clauses*, $\mathrm{GPeb}^{k\oplus}(G)$, are the clauses of the form

$$C \vee \rho(C) \qquad\text{and}\qquad C \vee \overline{\rho(C)}$$

for $C \in \mathrm{Peb}^{k\oplus}(G)$.

The $\mathrm{GPeb}^{k\oplus}(G)$ clauses depend on the choice of ρ; however, this is suppressed in the notation. $\mathrm{GPeb}^{k\oplus}(G)$ consists of $O(2^{3k}n)$ clauses.

Our definitions of $\mathrm{Peb}^{k\oplus}(G)$ and $\mathrm{GPeb}^{k\oplus}(G)$ differ somewhat from Urquhart's, but these differences are inessential and make no difference to asymptotic proof sizes.

Of course, the $\mathrm{Peb}^{k\oplus}(G)$ clauses are readily derivable from the $\mathrm{GPeb}^{k\oplus}(G)$ clauses by resolving on the guard literals as given by ρ. There are simple polynomial size regular resolution refutations of the $\mathrm{Peb}^{k\oplus}(G)$ clauses; hence there are polynomial size, but not regular, resolution refutations of the $\mathrm{GPeb}^{k\oplus}(G)$ clauses. Indeed, Urquhart (2011) proved that there are pointed graphs G with n vertices and values $k = k(n) = O(\log\log n)$, and functions ρ, such that regular resolution refutations of the $\mathrm{GPeb}^{k\oplus}(G)$ clauses require size $2^{\Omega(n/((\log n)^2 \log\log n))}$.

Theorem 10. *The guarded xor-ified pebbling tautology clauses* $\mathrm{GPeb}^{k\oplus}(G)$ *have polynomial size regRTI refutations, and thus polynomial size pool refutations.*

We make some simple observations about working with xor-ified clauses before proving Theorem 10.

Lemma 11. *Let u be a vertex in G. There is a tree-like regular refutation of the clauses in $x_u^{k\oplus}$ and $\overline{x}_u^{k\oplus}$ with $2^k - 1$ resolution inferences, height k, and 2^k leaf clauses. Its resolution variables are the variables $x_{u,i}$.*

Proof. This is immediate by inspection: the refutation consists of resolving on the literals $x_{u,i}$ successively for $i = 1, 2, \ldots, k$, giving a refutation of height k. The refutation corresponds to a complete binary decision tree over the k many variables $x_{u,i}$; the leaf clauses of the refutation are the members of $x_u^{k\oplus}$ and $\overline{x}_u^{k\oplus}$. ☐

The refutation of Lemma 11 can be viewed as being the "$k\oplus$-translation" of the proof

$$\frac{x_u \qquad \overline{x}_u}{\bot}$$

The next lemma describes a similar "$k\oplus$-translation" of a proof

$$\frac{C, x_u \qquad D, \overline{x}_u}{C, D}$$

Lemma 12. *Let u be a vertex in G, and let C and D be clauses which do not contain either x_u and $\overline{x}_u$. Then each clause of $(C \vee D)^{k\oplus}$ has a tree-like regular derivation from the clauses in $(C \vee x_u)^{k\oplus}$ and $(D \vee \overline{x}_u)^{k\oplus}$ in which the variables used as resolution variables are exactly the variables $x_{u,i}$. This derivation has $2^k - 1$ resolution inferences, height k, and 2^k leaf clauses.*

Proof. Fix a clause E from $(C \vee D)^{k\oplus}$; we must describe its derivation from clauses in $(C \vee x_u)^{k\oplus}$ and $(D \vee \overline{x}_u)^{k\oplus}$. Let E_C be the subclause of E which is from $C^{k\oplus}$, and let E_D the subclause of E which is from $D^{k\oplus}$. If C and D have non-empty intersection, E_C and E_D are not disjoint; however, in any event, $E = E_C \cup E_D$.

Form the refutation from Lemma 11. Then add E_C to every leaf clause from $x_u^{\oplus}$, add E_D to every leaf clause from $\overline{x}_u^{k\oplus}$, and add E to every non-leaf clause. This gives the desired derivation of E. ☐

Lemma 12 lets us further generalize the construction of $k\oplus$-translations of proofs. As a typical example, the next lemma gives the $k\oplus$-translation of the following derivation:

$$\dfrac{\dfrac{\overline{x}_u, \overline{x}_v, x_w \qquad x_u}{\overline{x}_v, x_w} \qquad x_v}{x_w}$$

Lemma 13. *Let w be a vertex of G, and u and v its predecessors. Then, each clause in $x_w^{k\oplus}$ has a dag-like regular resolution derivation P from the clauses in $x_u^{k\oplus}$, $x_v^{k\oplus}$, and $(\overline{x}_u \vee \overline{x}_v \vee x_w)^{k\oplus}$. This derivation contains $< 2^{2k}$ resolution inferences and resolves on the literals $x_{u,i}$ and $x_{v,i}$. In addition, the paths in P that lead to clauses in $x_v^{k\oplus}$ resolve on exactly the literals $x_{v,i}$.*

Lemma 13 follows by applying Lemma 12 twice. □

It is important to note that the left-to-right order of the leaves of the derivation of Lemma 13 can be altered by changing the left-to-right order of hypotheses of resolution inferences. In particular, given any leaf clause D of a refutation P, we can order the hypotheses of the resolution inferences so that D is the leftmost leaf clause. This will be useful when D needs to be learned.

Definition 14. $G{\restriction}w$ is the induced pointed subgraph of G with sink w and containing those vertices from which the vertex w is reachable. $G[w]$ is the subgraph of G obtained by making the vertex w a leaf by removing its incoming edges, and then removing those vertices from which the sink vertex of G is no longer reachable. Note that $G[w]$ is a pointed dag and has the same sink as G.

The vertex u is an *ancestor* of w if $u \neq w$ and $u \in G{\restriction}w$, i.e., if there is a path from u to w. We call u and v *independent ancestors* of w provided u, v, and w are distinct and $u \in (G{\restriction}w)[v]$ and $v \in (G{\restriction}w)[u]$. This means there is a path from u to w that does not contain v, and a path from v to w to that does not contain u. We write $G[u, v]$ for $G[u][v] = G[v][u]$.

More generally, let $\ell \geq 0$ and $u_1, \ldots, u_\ell, w$ be distinct vertices. We say $u_1, \ldots, u_\ell$ are *independent ancestors* of w, if for every $i \leq \ell$, there is a path from u_i to w that does not contain any u_j for $j \neq i$. We write $G[u_1, \ldots, u_\ell]$ for $G[u_1] \cdots [u_\ell]$. Note that the definition of $G[u_1, \ldots, u_\ell]$ is independent of the order of the u_i's.

Since G is a dag, it is possible for u and v to be independent ancestors of w, and also have u an ancestor of v or vice-versa.

The next lemma states that the polynomial size regular resolution refutations of the $\mathrm{Peb}^{k\oplus}(G)$ clauses also apply to subgraphs such as $G{\restriction}w$ and $(G{\restriction}w)[u_1, \ldots, u_\ell]$. We write $\mathrm{Peb}^{k\oplus}_{\alpha\beta}(G)$ to denote the $k\oplus$-translations of $\mathrm{Peb}(G)$ clauses of type (α) and (β), omitting the clauses of type (γ), and similarly for $\mathrm{GPeb}^{k\oplus}_{\alpha\beta}(G)$. To save space, we often write $\vec{u}$ instead of $u_1, \ldots, u_\ell$. For instance, in the next lemma, $((G{\restriction}w)[\vec{u}]) \setminus \{\vec{u}, w\}$ is shorthand for $((G{\restriction}w)[u_1, \ldots, u_\ell]) \setminus \{u_1, \ldots, u_\ell, w\}$.

Lemma 15. *Let w be a vertex of G. Let $\ell \geq 0$ and $u_1, \ldots u_\ell$ be independent ancestors of w. Then each clause of $(\overline{x}_{u_1} \vee \cdots \vee \overline{x}_{u_\ell} \vee x_w)^{k\oplus}$ has a regular resolution derivation from the clauses $\mathrm{Peb}^{k\oplus}_{\alpha\beta}((G{\restriction}w)[\vec{u}])$. The derivation uses only resolution variables of the form $x_{v,i}$ for $v \in ((G{\restriction}w)[\vec{u}]) \setminus \{\vec{u}, w\}$. The derivation is dag-like and has size $O(2^{3k}n)$ and height $O(kn)$.*

Proof. There is a simple regular dag-like derivation P of $\overline{x}_{u_1} \vee \cdots \vee \overline{x}_{u_\ell} \vee x_w$ from the clauses (α) and (β) of (non-xorified) $\mathrm{Peb}(G)$, of size $O(n)$ where n is the size of G. P proceeds by visiting vertices v in a depth-first traversal of $(G{\restriction}w)[\vec{u}]$ and deriving some subclause C_v of $\overline{x}_{u_1} \vee \cdots \vee \overline{x}_{u_\ell} \vee x_v$: the subclause C_v contains x_v and those x_{u_i}'s such that there is a path from u_i to v that contains no other u_j. For v a leaf distinct from the u_j's, C_v is just x_v and also a Peb(G) clause of type (α). If v has immediate predecessors v_1 and v_2, then C_v is formed by resolving C_{v_1} and C_{v_2} against the Peb(G) clause $\overline{x}_{v_1} \vee \overline{x}_{v_2} \vee x_v$ of type (β).

Now let U be some clause in $(\overline{x}_{u_1} \vee \cdots \vee \overline{x}_{u_\ell})^{k\oplus}$ and W some clause in $x_w^{k\oplus}$. We need to give a derivation of $U \vee W$. We claim that a $k\oplus$-translation $P^{k\oplus}$ of P forms the desired derivation. For this, each clause C_v of P is translated into 2^{k-1} many clauses C_X where $X \in x_v^{k\oplus}$. Each C_X is a subclause of $U \vee X$; namely the subclause which omits the literals $x_{u_{i,j}}$ and $\overline{x}_{u_{i,j}}$ of U when $\overline{x}_{u_i} \notin C_v$. For v_1 and v_2 the two immediate ancestors of v as before, and $X \in x_v^{k\oplus}$, the clause C_X is derived in $P^{k\oplus}$ from the 2^{2k-2} many clauses C_{X_1} and C_{X_2} where $X_i \in (x_{v_i})^{k\oplus}$ for $i = 1, 2$. By Lemma 13, each such subderivation in $P^{k\oplus}$ has height $\leq 2k$ and size $\leq 2^{2k}$, and resolves on exactly the variables $x_{v_1,j}$ and $x_{v_2,j}$.

The result is that a variable $x_{v,j}$ is resolved on in $P^{k\oplus}$ precisely when x_v is resolved on in P. And, since P has size $O(n)$, $P^{k\oplus}$ has size $O(2^{3k}n)$ and height $O(kn)$. □

Proof. (of Theorem 10.) We will construct a series of "LR partial refutations", denoted R_0, R_1, $R_2, \ldots$; this process eventually terminates with a pool resolution (regRTL) refutation of $\mathrm{GPeb}^{k\oplus}(G)$. 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 labeled with "unfinished clauses" instead of with valid learned clauses or initial clauses from $\mathrm{GPeb}^{k\oplus}(G)$.

An LR partial refutation R is a tree with nodes labeled with clauses that form a correct regRTI resolution refutation (and thus a correct pool resolution refutation), except at the unfinished clauses at leaves. Furthermore, it must satisfy the following conditions:

a. R_t is a tree of nodes labeled with clauses. The root is labeled with the empty clause. Each non-leaf node in R_t has a left child and a right child, and the clauses labeling these nodes form a valid resolution inference.

b. Each leaf of R_t is either "finished" or "unfinished". Each finished node leaf L is labeled with either a clause from $\mathrm{GPeb}^{k\oplus}(G)$ or with a clause that was derived by an input subderivation of R_t to the left of L in the post-order. The input subderivation may not contain any unfinished leaves.

c. Each unfinished leaf is labeled with a clause $C \in E^{k\oplus}$ where E is a clause of the form $\overline{x}_{u_1} \vee \cdots \vee \overline{x}_{u_\ell} \vee x_w$ with $\ell \geq 0$ and $u_1, \ldots, u_\ell$ independent ancestors of w. Furthermore C^{pool} contains no literal $x_{v,i}$ with $v \in (G{\restriction}w)[\vec{u}] \setminus \{\vec{u}, w\}$.

We introduce a new notational convention to describe (sub)clauses in R_t. For w a vertex in G, the notation W or W' denotes a clause in $x_w^{k\oplus}$, and $\overline{W}$ or $\overline{W}'$ denotes a clause in $\overline{x}_w^{k\oplus}$. The notation $\overline{W}$ or $\overline{W}'$ in no way denotes the negation of W or W'; instead, they are names of clauses, with the overline meant only to serve as a reminder of the semantic meaning.

The initial LR-partial refutation R_0 is formed as follows. Let Q be the refutation obtained as the $k\oplus$-translation of the inference

$$\frac{\overline{x}_t \qquad x_t}{\bot}$$

as given by Lemma 11, where t is the sink of G. There are 2^k leaf clauses of Q: half of them are labeled with clauses $\overline{T} \in \overline{x}_t^{k\oplus}$ and the other half are labeled with clauses $T \in x_t^{k\oplus}$. Form R_0 from Q by replacing each leaf clause $\overline{T} \in \overline{x}_t^{k\oplus}$ with a derivation

$$\frac{\overline{T}, \rho(\overline{T}) \qquad \overline{T}, \overline{\rho(\overline{T})}}{\overline{T}}$$

resolving on the guard literal $\rho(\overline{T})$. These inferences are regular, since $\rho(\overline{T})$ is not an $x_{t,i}$. The clauses $\overline{T}, \rho(\overline{T})$ and $\overline{T}, \overline{\rho(\overline{T})}$ are in $\mathrm{GPeb}^{k\oplus}(G)$ and hence are finished clauses. The other leaf clauses, of the form $T \in x_t^{k\oplus}$, satisfy condition c. with $T = C \in x_t^{k\oplus}$ and $\ell = 0$; these T's are unfinished clauses in R_0.

For the inductive step $t \geq 0$, the LR partial refutation R_t will be transformed into R_{t+1}. The goal is to replace one unfinished leaf in R_t, either by a derivation containing only finished leaves, or by a derivation which learns one more $\mathrm{Peb}^{k\oplus}(G)$ clause while adding only polynomially many more unfinished leaves.

Consider the leftmost unfinished leaf of R_t. By condition c., its clause C will have the form $\overline{U}_1, \ldots, \overline{U}_\ell, W$ where $\ell \geq 0$, $\overline{U}_i \in \overline{x}_{u_i}^{k\oplus}$ for $i \leq \ell$, and $W \in x_w^{k\oplus}$. By Lemma 15, there is a dag-like regular refutation P of C from the clauses of $\mathrm{Peb}_{\alpha\beta}^{k\oplus}((G{\restriction}w)[\vec{u}])$. We wish to convert P (if possible) into a derivation of C from the clauses of $\mathrm{GPeb}_{\alpha\beta}^{k\oplus}((G{\restriction}w)[\vec{u}])$ and the already learned clauses of R_t. Consider a particular leaf clause D of P, so $D \in \mathrm{Peb}_{\alpha\beta}^{k\oplus}((G{\restriction}w)[\vec{u}])$: each such D needs to be handled in some way that makes P a valid derivation. There are four cases to consider:

(*i*) If the clause D is already learned as an input lemma in R_t to the left of C, then D may be used in P as is.

For the remaining cases, assume D has not been learned as an input lemma.

(*ii*) Let $y = \rho(D)$. If either y or $\overline{y}$ is a member of C^{pool}, then add that literal to D and to every clause below D until reaching the first clause where it appears. This replaces D with one of the $\mathrm{GPeb}_{\alpha\beta}^{k\oplus}((G{\restriction}w)[\vec{u}])$ clauses $D \vee y$ or $D \vee \overline{y}$. By construction, it preserves the validity of the resolution inferences of R_t as well as the regularity property.

(*iii*) Suppose cases (*i*) and (*ii*) do not apply and that y is not used in P as a resolution variable below D. In this case, replace D by a resolution inference deriving D from $D \vee y$ and $D \vee \overline{y}$. This preserves the regularity of the derivation. It also makes D a learned clause.

It is possible that C itself is a $\mathrm{Peb}_{\alpha\beta}^{k\oplus}(G)$ clause. If so, then $C = D$ and P is the trivial derivation containing only C, and one of cases (*i*)-(*iii*) holds.

If all leaf clauses D of P can be treated by cases (*i*)-(*iii*), then we have successfully transformed P into a (still dag-like) derivation P' which satisfies regularity and in which leaf clauses are from $\mathrm{GPeb}^{k\oplus}(G)$ or already learned as input lemmas in R_t. By a result of Buss et al. (2008, Thm. 3.3), P' can be converted in a regRTI proof P'' of the same conclusion as P, preserving the regularity conditions, and with the size of P' bounded by twice the product of the size of P and the height of P. Therefore, the size of P'' is $O((2^{3k}n)(kn)) =$

$O(k2^{3k}n^2)$. Form R_{t+1} by replacing the clause C in R_t with the derivation P''. R_{t+1} satisfies conditions a.-c., and has one fewer unfinished clauses than R_t.

However, if even one leaf clause D of P fails cases (*i*)-(*iii*), then a completely different construction is used to form R_{t+1}. Fix some leaf clause D which of P does not fall into cases (*i*)-(*iii*). The unfinished clause C of R_t will be replaced by a small derivation of C which learns D (in its leftmost inference), and which adds up to $O(2^{3k})$ new unfinished clauses in R_{t+1}.

The leaf clause D is the $k\oplus$-translation of an (α) or (β) clause of $\mathrm{Peb}^{k\oplus}(G)$ and thus either has the form $E \in x_e^{\oplus}$ for some source e in G or has the form $\overline{A}, \overline{B}, E$ where $\overline{A} \in \overline{x}_a^{k\oplus}$, $\overline{B} \in \overline{x}_b^{k\oplus}$, and $E \in x_e^{k\oplus}$ for a, b, and e vertices in $G\restriction w$ with a and b the two predecessors of e in G. Without loss of generality, b is not an ancestor of a in G; otherwise interchange a and b. There are two cases depending on whether D is E or is $\overline{A}, \overline{B}, E$.

First suppose D is $E \in x_e^{k\oplus}$ for e a source node in G. We claim that $e, u_1, \ldots, u_\ell$ are independent ancestors of w. As already remarked, P is not the trivial derivation since otherwise the cases (*i*)-(*iii*) would hold; therefore e is not equal to w. Consequently, E is not a subclause of the final clause $C = \overline{U}_1, \ldots \overline{U}_\ell, W$ of P. Hence some $x_{e,i}$ is resolved on in P. Since P was formed using Lemma 15, e therefore cannot equal any u_i. Since $e \in (G\restriction w)[\vec{u}]$, there must exist a path from e to w that avoids all the nodes u_i. Thus, since e is a source node, the vertices $e, u_1, \ldots, u_\ell$ are independent ancestors of w.

To form R_{t+1}, first replace the derivation P by the $k\oplus$-translation of the following:

$$\frac{x_e \qquad \overline{x}_e, \overline{U}_1, \ldots, \overline{U}_\ell, W}{\overline{U}_1, \ldots, \overline{U}_\ell, W} \tag{2}$$

Note that (2) contains a blend of variables from $\mathrm{Peb}(G)$ (non-xorified) and from $\mathrm{Peb}^{k\oplus}(G)$ (xor-ified). However, we can still form its $k\oplus$-translation Q: the leaf clauses of Q are the 2^k clauses of the form $E' \in x_e^{k\oplus}$ and of the form $\overline{E}', \overline{U}_1, \ldots, \overline{U}_\ell, W$ for $\overline{E}' \in \overline{x}_e^{k\oplus}$. By choosing the appropriate left-to-right order for the hypotheses in Q, we arrange for $D = E$ to be the leftmost leaf clause of Q. Let $y = \rho(D)$. Then y is not one of the variables $x_{e,i}$, nor is y or $\overline{y}$ in C^{pool} since condition (*ii*) does not hold for D. Therefore, the regularity condition is preserved when we modify Q by replacing D with

$$\frac{D, \rho(D) \qquad D, \overline{\rho(D)}}{D} \tag{3}$$

Form R_{t+1} from R_t by replacing C with the modified Q. This causes D to become learned as an input lemma in R_{t+1}. The other leaf clauses of Q all satisfy condition c. in R_{t+1} and thus become unfinished clauses of R_{t+1}: they are all to the right of D. This adds $< 2^k$ new unfinished clauses to R_{t+1}. The number of inferences in the derivation structure is less than 2^k.

Second suppose D is $\overline{A}, \overline{B}, E$, where a and b are the predecessors of e in G. Suppose for the moment that $e \neq w$, and that the nodes $u_1, \ldots, u_\ell$ are distinct from a and b. By the same reasoning as in the previous case, e is not equal to any u_i. We begin by replacing the derivation P with the $k\oplus$-translation Q of:

$$\dfrac{\dfrac{\dfrac{\overline{x}_a, \overline{x}_b, x_e \qquad \mathbb{U}_a, x_a}{\mathbb{U}_a, \overline{x}_b, x_e} \qquad \mathbb{U}_b, x_b}{\mathbb{U}_a, \mathbb{U}_b, x_e} \qquad \mathbb{U}_w, \overline{x}_e, W}{\overline{U}_1, \ldots, \overline{U}_\ell, W} \tag{4}$$

where $\mathbb{U}_a$, $\mathbb{U}_b$ and $\mathbb{U}_w$ are (possibly empty) sets of literals that form a partition of the set $\{\overline{U}_1, \ldots, \overline{U}_\ell\}$. We must define $\mathbb{U}_a$, $\mathbb{U}_b$, and $\mathbb{U}_w$ so that (the $k\oplus$-translation of) the clause $\mathbb{U}_a, x_a$, the clause $\mathbb{U}_b, x_b$ and the clause $\mathbb{U}_w, \overline{x}_e, W$ each fulfill the condition c. in order to be legitimate "unfinished leaves".

Since $u_1, \ldots, u_\ell$ are independent ancestors of w, we can fix paths $\pi_1, \ldots, \pi_\ell$ in G such that π_i is a path from u_i to w and such that π_i does not contain u_j for any $j \neq i$. Call π_i an "a-path" if it contains the node a. Call π_i a "b-path if it contains b but does not contain a. Finally, call π_i a "w-path" if it contains neither a nor b. Then, define $\mathbb{U}_a$ to be the set of U_i's such that π_i is an "a-path. Likewise, let $\mathbb{U}_b$ (respectively, $\mathbb{U}_w$) be the set of U_i's such that π_i is a b-path (respectively a w-path). Clearly, $\mathbb{U}_a$, $\mathbb{U}_b$ and $\mathbb{U}_w$ form a partition of $\{U_1, \ldots, U_\ell\}$. Also, the u_i's for U_i in $\mathbb{U}_a$ form a set of independent ancestors of a, so $\mathbb{U}_a, x_a$ satisfies condition c. Likewise, the clauses $\mathbb{U}_b, x_b$ and $\mathbb{U}_w, \overline{x}_e, W$ also satisfy condition c. To prove the latter, note that any path from u_i to w that contains e must contain at least one of a or b.

The $k\oplus$-translation Q of (4) has D as its leftmost leaf clause. Let $y = \rho(D)$. Then y is not one of the variables $x_{a,i}, x_{b,i}, x_{e,i}$, nor is y or $\overline{y}$ in C^{pool} since condition (*ii*) does not hold for D. Therefore, the regularity condition is preserved when we modify Q by replacing D with (3). Form R_{t+1} from R_t by replacing C with the modified Q. This causes D to become learned as an input lemma in R_{t+1}. As previously argued, the other leaf clauses of Q have become valid unfinished clauses that satisfy condition c. R_{t+1} gains $< 2^{3k}$ new inferences, and less than $3 \cdot 2^k$ new unfinished clauses.

We still have to consider the cases where $u_1, \ldots, u_\ell$, a and b are not distinct. There are 2ℓ (very similar) cases where only one of the u_i's is in $\{a, b\}$. For instance, suppose that $u_1 = a$ and no u_i is equal to b. We claim that A is the same clause as U_1. To prove this, note that if A is different from U_1, then there is a literal $x_{u_1,j}$ or $\overline{x}_{u_1,j}$ appearing in A which appears in negated form $\overline{x}_{u_1,j}$ or $x_{u_1,j}$ (respectively) in U_1. This implies that the derivation P uses $x_{u_1,j}$ as a resolution variable, contradicting the fact that P was formed via Lemma 15. With $A = U_1$, we form Q as the $k\oplus$-translation of

$$\dfrac{\dfrac{\overline{U}_1, \overline{x}_b, x_e \qquad \mathbb{U}_b, x_b}{\mathbb{U}_b, \overline{U}_1, x_e} \qquad \mathbb{U}_w, \overline{x}_e, W}{\overline{U}_1, \ldots, \overline{U}_\ell, W}$$

Here $\mathbb{U}_b$ and $\mathbb{U}_w$ are defined as above (and $\mathbb{U}_a$ equals $\{U_1\}$). Order Q so that D is its leftmost leaf clause, and then form R_{t+1} as in the previous paragraph.

The case where there are $i, j \leq \ell$ and $i \neq j$, such that $u_i = a$ and $u_j = b$, the proof structure is even simpler: Suppose $u_1 = a$ and $u_2 = b$. Similarly to the previous case, A and B must be the same as U_1 and U_2, respectively. Then let Q be the $k\oplus$-translation of

$$\dfrac{\overline{U}_1, \overline{U}_2, x_e \qquad \overline{x}_e, \overline{U}_3, \ldots, \overline{U}_\ell, W}{\overline{U}_1, \ldots, \overline{U}_\ell, W}$$

(so $\mathbb{U}_w = \{U_3, \ldots, U_\ell\}$) and proceed as before.

Finally, consider the case $w = e$. Clearly, each U_i is an ancestor of either a or b. Therefore, each U_i is in $\mathbb{U}_a$ or $\mathbb{U}_b$. The refutation R_{t+1} is formed from the proof structure as in (4), but omitting the last inference.

This concludes the construction of R_{t+1} from R_t. The process of constructing R_t's halts once there are no remaining unfinished clauses, and yields a final refutation R which is a valid regRTI refutation of the $\mathrm{GPeb}^{k\oplus}(G)$ clauses.

We need to bound the size of the refutation R. First consider how R_{t+1} is formed from R_t. In cases (*i*)-(*iii*), an unfinished leaf is completely handled without adding any new unfinished leaves. In these cases, at most $O(k2^{3k}n^2)$ many new clauses are introduced in R_{t+1}. In case (*iv*), a new $\mathrm{Peb}^{k\oplus}(G)$ clause is learned as an input lemma while adding only $O(3 \cdot 2^k)$ many new unfinished leaves in R_{t+1} (and only $O(2^{3k})$ new clauses).

Even though there are exponentially many potential unfinished clauses, the case (*iv*) construction can occur at most polynomially many times because there are only polynomially many $\mathrm{Peb}^{k\oplus}(G)$ clauses to be learned. Indeed, there are only $< n2^{3(k-1)}$ many $\mathrm{Peb}^{k\oplus}(G)$ clauses. Therefore, at most $O(n2^{3(k-1)} \cdot 3 \cdot 2^k)$ many distinct unfinished leaf clauses can appear during the construction of R. Consequently, cases (*i*)-(*iii*) occur only this many times. Therefore, the total number of clauses in R is bounded by $O(n2^{3(k-1)} \cdot 3 \cdot 2^k \cdot k2^{3k}n^2) = O(2^{7k}n^3)$. Thus the size of R is polynomially bounded by the size of the $\mathrm{GPeb}^{k\oplus}(G)$ clauses; in fact, it is bounded by a degree three polynomial.

This completes the proof of Theorem 10. □

4. Guarded Graph Tautologies

We define various graph tautologies, sometimes also called “ordering principles”. They 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 will thus always adopt the simplifying convention that $x_{i,j}$ and $\overline{x}_{j,i}$ are the identical literal, i.e., only the variables $x_{i,j}$ for $i < j$ actually exist, and $x_{j,i}$ for $j < i$ is just a notation for $\overline{x}_{i,j}$, and $\overline{x}_{j,i}$ stands for $x_{i,j}$. This identification makes no essential difference to the complexity of proofs of the tautologies, but it reduces the number of literals and clauses, and simplifies the definitions. In particular, it means there are no axioms for the antisymmetry or totality of $\prec$.

The following GT_n clauses are based on the tautologies defined by Krishnamurthy (1985). These tautologies, or similar ones, have also been studied by Stålmarck (1996), Bonet and Galesi (2001), Segerlind, Buss, and Impagliazzo (2004), Beckmann and Buss (2005), Van Gelder (2006), Alekhnovich et al. (2007) and Johannsen (2009).

Definition 16. 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 $i < n$.

($\beta_\emptyset$) 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. For this reason Van Gelder (2005) uses the name “no triangles” (NT) for a similar principle.

The next definition is due to Alekhnovich et al. (2007), 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 $\{r(i,j,k), s(i,j,k)\} \not\subset \{i,j,k\}$. In addition, w.l.o.g., $r(i,j,k) = r(j,k,i) = r(k,i,j)$, and similarly for s.

Definition 17. 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 clauses*, GGT_n, consist of:

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

($\beta'_\emptyset$) 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 $s = s(i,j,k)$.

Note that the GGT_n clauses depend on the functions r and s; this is suppressed in the notation. Our first main result for the guarded graph tautologies is:

Theorem 18. *The guarded graph tautology clauses* GGT_n *have polynomial size pool resolution (regRTL) refutations.*

The proof of Theorem 18 will construct pool refutations in the form of regular tree-like refutations with lemmas. A key part of this is learning transitive closure clauses that are derived using resolution on the guarded transitivity clauses of GGT_n. A slightly modified construction, that uses a result of Buss et al. (2008), gives instead tree-like regular resolution refutations with *input* lemmas. This will establish the following:

Theorem 19. *The guarded graph tautology clauses* GGT_n *have polynomial size, tree-like regular resolution refutations with input lemmas (regRTI refutations).*

As discussed in the introduction, Theorem 19 and a result of Buss et al. (2008) together imply the GGT_n clauses can be shown unsatisfiable by non-greedy polynomial size CDCL. This follows via the mentioned theorem of Buss et al. (2008, Thm. 5.6), since the refutations of GGT_n are regRTI, and hence regWRTI, proofs in the sense of Buss et al.. Theorem 31 of Section 5 will improve on this by giving greedy CDCL refutations.

Theorem 19 is strictly stronger than Theorem 18, but we find it convenient to prove Theorem 18 first.

4.1 Resolution Refutations for Guarded Graph Tautologies

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

Theorem 20. (Stålmarck, 1996; Bonet & Galesi, 2001; Van Gelder, 2006) *The sets* GT_n *have regular resolution refutations* P_n *of polynomial size* $O(n^3)$.

The proofs of Theorems 18 and 19 use the refutation P_n as a "black box": the only property needed is that the P_n's are regular and polynomial size. Section 5 will need to use the details of P_n however.

Proof. (Proof sketch for Theorem 20.) For $\ell, k \in [n]$, define the clause $\mathrm{Tot}_{k,\ell}$ to be

$$\mathrm{Tot}_{k,\ell} := \bigvee_{j \in [k+1] \setminus \{\ell\}} x_{j,\ell},$$

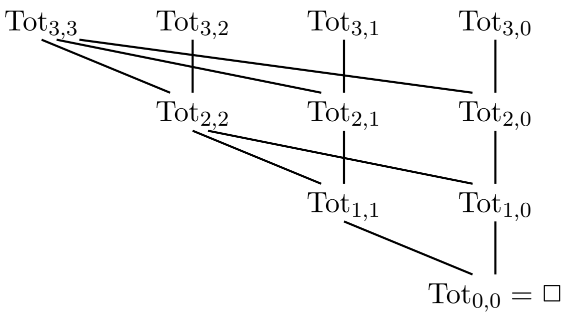


Figure 1: Main structure of the regular refutation P_n of the GT_n clauses, with $n = 4$. Inferences that resolve against transitivity clauses $T_{i,j,k}$ are not shown: the slanted line from $\mathrm{Tot}_{k+1,k+1}$ to $\mathrm{Tot}_{k,\ell}$ represents k many resolution inferences against the clauses $T_{\ell,i,k+1}$ for $i \in [k+1] \setminus \{\ell\}$. Each subderivation with hypotheses $\mathrm{Tot}_{k,k}$ and $\mathrm{Tot}_{k,\ell}$ and conclusion $\mathrm{Tot}_{k-1,\ell}$ is an input derivation.

which expresses the condition that ℓ has a predecessor $j \leq k$. Of course, the clauses $\mathrm{Tot}_{n-1,\ell}$, for $\ell = 0, 1, \ldots, n-1$, are the initial clauses $(\alpha_\emptyset)$ of GT_n. Note that $\mathrm{Tot}_{0,0}$ is the empty clause.

As pictured in Figure 1, P_n proceeds by deriving $\mathrm{Tot}_{k,\ell}$ from $\mathrm{Tot}_{k+1,k+1}$ and $\mathrm{Tot}_{k+1,\ell}$ for $k = n-2, \ldots, 0$ and $\ell \leq k$, until deriving the empty clause $\mathrm{Tot}_{0,0}$. The first part of the derivation of $\mathrm{Tot}_{k,\ell}$ resolves $\mathrm{Tot}_{k+1,k+1}$ successively against the k many transitivity clauses $T_{\ell,i,k+1}$ for $i \in [k+1] \setminus \{\ell\}$. By the convention that $x_{i,\ell}$ and $\overline{x}_{\ell,i}$ are the same literal, $T_{\ell,i,k+1}$ is the clause $x_{i,\ell}, \overline{x}_{i,k+1}, \overline{x}_{k+1,\ell}$. The resolution with $T_{\ell,i,k+1}$ uses $x_{i,k+1}$ as the resolution literal and has the effect of adding $\overline{x}_{k+1,\ell}$ to the clause, and replacing $x_{i,k+1}$ with $x_{i,\ell}$. Thus, the conclusion of these k resolution steps is

$$\overline{x}_{k+1,\ell} \vee \bigvee_{i \in [k+1] \setminus \{\ell\}} x_{i,\ell}.$$

One final resolution against $\mathrm{Tot}_{k+1,\ell}$ on the literal $x_{k+1,\ell}$ yields $\mathrm{Tot}_{k,\ell}$ as desired. The regularity of P_n is evident by inspection. □

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 $(\beta'_\emptyset)$. This however destroys the regularity property, and as already discussed, no polynomial size regular refutations exist for GGT_n (Alekhnovich et al., 2007).

As usual, a *partial order* $\prec$ on $[n]$ is an antisymmetric, transitive binary relation on $[n]$. We will be primarily interested in "bipartite" partial orders, which are partial orders that do not have any chain of inequalities $x \prec y \prec z$.

Definition 21. A *bipartite partial order* is a binary relation π on $[n]$ such that the domain and range of π do not intersect. We write $x \prec_\pi y$ for $(x, y) \in \pi$. The set of π-minimal elements is denoted M_π.

The righthand side of Figure 2 shows an example. The bipartiteness of π arises from the fact that M_π and $[n] \setminus M_\pi$ partition $[n]$ into two sets. If $i \prec_\pi j$, then $i \in M_\pi$ and $j \notin M_\pi$.

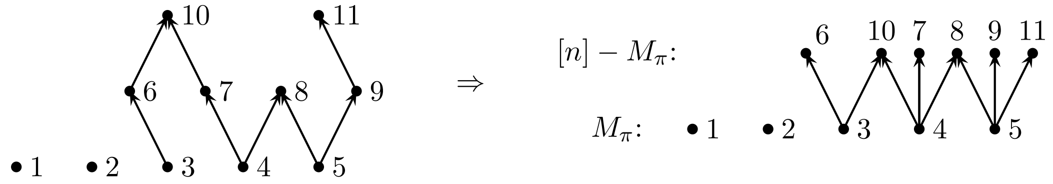


Figure 2: Example of a dag (left) and its associated bipartite partial order (right).

In addition, M_π contains the isolated points of π. It is permitted for $\prec$ to be the empty relation, and then $M_\pi = [n]$.

A (bipartite) partial order on $[n]$ can also be viewed as a directed acyclic graph (dag) $G = ([n], \pi)$; note that $[n]$ is the set of vertices, and π is the set of edges of G. Conversely, we define a mapping from an arbitrary dag $G = ([n], \tau)$ to an associated bipartite partial order π:

Definition 22. Let $G = ([n], \tau)$ be a dag. We write $x \prec_\tau^+ y$ to denote that G contains a path of length ≥ 1 from x to y. The bipartite partial order π *associated with* G is defined by letting $x \prec_\pi y$ hold precisely when x is τ-minimal in G (that is, x has indegree 0) and $x \prec_\tau^+ y$.

Note that the definition does not require τ to be transitive. It is easy to check that the π associated with τ is in fact 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.

We define the graph tautology clauses $\mathrm{GT}_{\pi,n}$ relative to π as follows.

Definition 23. Let π be a bipartite partial order on $[n]$. Then $\mathrm{GT}_{\pi,n}$ contains:

(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 Figure 3 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 Figure 3.)

The set $\mathrm{GT}_{\pi,n}$ is satisfiable if π is nonempty. As an example, there is 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, if π is applied as a restriction, then $\mathrm{GT}_{\pi,n}$ becomes unsatisfiable. That is to say, 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 described in the next lemma.

Definition 24. For π a bipartite partial order, the clause $(\bigvee \overline{\pi})$ is

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

Lemma 25. *Let π be a bipartite partial order on $[n]$. Then there is a regular derivation P_π of $(\bigvee \overline{\pi})$ 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 $i \not\prec_\pi k$.

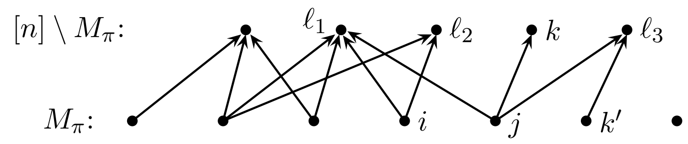


Figure 3: A bipartite partial order π is pictured, with the ordered pairs of π shown as directed edges. (For instance, $j \prec_\pi k$ holds.) The set M_π is the set of minimal vertices. The nodes i, j, k shown are an example of nodes used for a transitivity axiom $\overline{x}_{i,j} \vee \overline{x}_{j,k} \vee \overline{x}_{k,i}$ of type (γ). The nodes i, j, k' are an example of the nodes for a transitivity axiom of type (β).

Lemma 25 implies that if π is the bipartite partial order associated with a dag τ, then 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 the P_π of Lemma 25 is the same as the refutation of GT_n of Theorem 20.

Proof. By renumbering the vertices, we can assume w.l.o.g. that $M_\pi = \{0, \ldots, m-1\}$. For each $k \geq m$, there is at least one value of j such that $j \prec_\pi k$: let J_k be an arbitrary such value j. Note $J_k < m$.

Fix $i \in M_\pi$; that is, $i < m$. Recall that the clause of type (α) in $\mathrm{GT}_{\pi,n}$ for i is $\bigvee_{j \neq i} x_{j,i}$. We resolve this clause successively, for each $k \geq m$ such that $i \not\prec_\pi k$, against the clauses $T_{i,J_k,k}$ of type (γ)

$$\overline{x}_{i,J_k} \vee \overline{x}_{J_k,k} \vee \overline{x}_{k,i}$$

using resolution variables $x_{k,i}$. (Note that $J_k \neq i$ since $i \not\prec_\pi k$.) This yields a clause $\mathrm{Tot}^\pi_{m-1,i}$:

$$\bigvee_{\substack{k \geq m \\ i \not\prec_\pi k}} \overline{x}_{i,J_k} \;\vee\; \bigvee_{\substack{k \geq m \\ i \not\prec_\pi k}} \overline{x}_{J_k,k} \;\vee\; \bigvee_{\substack{k \geq m \\ i \prec_\pi k}} x_{k,i} \;\vee\; \bigvee_{\substack{k < m \\ k \neq i}} x_{k,i}.$$

The first two disjuncts shown above come from the side literals of the clauses $T_{i,J_k,k}$; the last two disjuncts come from the literals in $\bigvee_{j \neq i} x_{j,i}$ which were not resolved on. Since a literal $\overline{x}_{i,J_k}$ is the same literal as $x_{J_k,i}$ and since $J_k < m$, the literals in the first disjunct are also contained in the fourth disjunct. Thus, eliminating duplicate literals, $\mathrm{Tot}^\pi_{m-1,i}$ is equal to the clause

$$S^\pi_i \vee \mathrm{Tot}_{m-1,i} \;:=\; \bigvee_{\substack{k \geq m \\ i \not\prec_\pi k}} \overline{x}_{J_k,k} \;\vee\; \bigvee_{\substack{k \geq m \\ i \prec_\pi k}} x_{k,i} \;\vee\; \bigvee_{\substack{k < m \\ k \neq i}} x_{k,i}. \tag{5}$$

where S^π_i is defined to equal the first two big disjuncts in the lefthand side of the equation, and $\mathrm{Tot}_{m-1,i}$ is the same as before, namely the last big disjunct on the righthand side.

Repeating this process, we obtain derivations of the clauses $\mathrm{Tot}^\pi_{m-1,i}$ for all $i < m$. Their subclauses $\mathrm{Tot}_{m-1,i}$ are the same as the ($\alpha_\emptyset$) clauses in GT_m. Thus, the clauses $\mathrm{Tot}^\pi_{m-1,i}$ give all ($\alpha_\emptyset$) clauses of GT_m, but with S^π_i's added in as side literals. Moreover, the clauses of type (β) in $\mathrm{GT}_{\pi,n}$ are exactly the transitivity clauses of GT_m. All these clauses can be

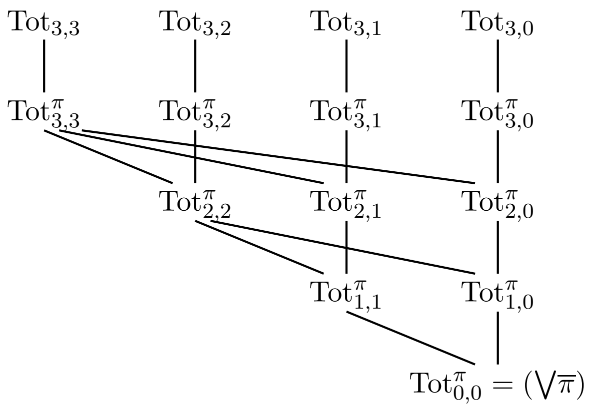


Figure 4: Main structure of the regular derivation P_π, with $m = 4$. The initial clauses are GT_m clauses. Inferences that resolve against transitivity clauses $T_{i,j,k}$ are not shown: slanted lines and the top row of vertical lines represent multiple resolution inferences against transitivity clauses.

combined exactly as in the refutation of GT_m described in Theorem 20, but carrying along extra side literals S^π_i; namely carrying along literals $\overline{x}_{J_k,k}$ for $J_k \prec_\pi k$, and $\overline{x}_{i,k}$ for $i \prec_\pi k$. Since the refutation of GT_m uses all of its transitivity clauses and since each $\overline{x}_{J_k,k}$ literal is the same as an $\overline{x}_{i,k}$ with $i \prec_\pi k$, this yields a resolution derivation P_π of the clause

$$\{\overline{x}_{i,k} : i \prec_\pi k\}.$$

This is the clause $(\bigvee\overline{\pi})$ as desired.

The just-constructed derivation P_π of $(\bigvee\overline{\pi})$ has the structure shown in Figure 4: the clauses $\mathrm{Tot}^\pi_{k,i}$ are equal to

$$\mathrm{Tot}^\pi_{k,i} \;=\; \begin{cases} S^\pi_i \vee \mathrm{Tot}_{k,i} & \text{if } k = m-1 \\ S^\pi_i \vee S^\pi_{m-1} \vee \mathrm{Tot}_{k,i} & \text{if } i < k < m-1 \\ \bigvee_{j=i}^{m-1} S^\pi_j \vee \mathrm{Tot}_{k,i} & \text{if } i = k \end{cases} \tag{6}$$

To show that P_π is regular, note that the first parts of P_π deriving the clauses $\mathrm{Tot}^\pi_{i,m}$ are regular by construction, and they use resolution only on variables $x_{k,i}$ with $i < m \le k$, and $i \not\prec_\pi k$. The remaining part of P_π is also regular by Theorem 20, and uses resolution only on variables $x_{i,j}$ with $i, j \le m$. □

4.2 Pool and w-Resolution Refutations for $\mathbf{GGT}_n$

We now prove Theorem 18, and then indicate the minor changes needed to prove Theorem 19.

Proof. (Theorem 18) We again construct a finite sequence of "LR partial refutations", denoted $R_0, R_1, R_2, \ldots$. This terminates after finitely many steps with the desired pool (regRTL) refutation R of GGT_n. Each LR partial refutation R_t will be a correct pool resolution refutation, except possibly for the presence of "unfinished clauses" at leaves. Unfinished clauses will correspond to bipartite partial orders. Each R_t will 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 each clause C occurring in R, the set of ordered pairs $\tau(C)$ is defined as $\tau(C) = \{\langle i,j\rangle : \overline{x}_{i,j} \in C^{\mathrm{pool}}\}$. In many cases, $\tau(C)$ will be a dag, but this is not always true. For instance, if C is a transitivity axiom, then $\tau(C)$ has a 3-cycle and is not a dag.

d. Leaves are either "finished" or "unfinished". 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 post-order traversal of R.

e. For an unfinished leaf labeled with clause C, the set $\tau(C)$ is a dag. Furthermore, letting π be the bipartite partial order associated with $\tau(C)$, the clause C is equal to $(\bigvee\overline{\pi})$.

Property e. is particularly crucial and is novel to 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 regularity property of pool resolution proofs. This can mean that S will instead be a derivation of a clause C' such that $C \subseteq C' \subseteq C^{\mathrm{pool}}$. The condition $C' \subseteq C^{\mathrm{pool}}$ 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. Since $C' \subseteq C^{\mathrm{pool}}$, such a clause exists. The fact that $C' \supseteq C$ implies that enough literals are present for the derivation to use only (non-degenerate) resolution inferences — by virtue of the fact that 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 $\tau(\emptyset) = \emptyset$. Thus R_0 is a valid LR partial refutation.

For the induction step, R_t has been constructed already. Let C be the leftmost unfinished clause in R_t. R_{t+1} will be formed by replacing C with an (LR-partial) derivation S of some clause C' such that $C \subseteq C' \subseteq C^{\mathrm{pool}}$.

We need to describe S. Let π be the bipartite partial order associated with $\tau(C)$, and consider the derivation P_π from Lemma 25. Since C is $(\bigvee\overline{\pi})$ by condition e., the final line of P_π is the clause C. The intuition is that we would like to let S be P_π. The first difficulty with this is that P_π is dag-like, and the LR-partial 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, the derivation P_π uses the transitivity clauses of GT_n as initial clauses instead of the guarded transitivity clauses of GGT_n. The transitivity clauses $T_{i,j,k} := \overline{x}_{i,j} \vee \overline{x}_{j,k} \vee \overline{x}_{k,i}$ in P_π are handled one at a time as described below. There are four separate constructions: case (i) requires no change to P_π; cases (ii) and (iii) require small changes; but in the fourth case, the subproof P_π is abandoned in favor of "learning" the transitivity clause.

By the remark after Lemma 25, no literal in C^{pool} is used as a resolution literal in P_π.

(i) Suppose a transitivity clause $T_{i,j,k}$ of P_π already appears earlier in R_t, that is, to the left of C in the post-order. Then $T_{i,j,k}$ is already learned, and can be used freely in P_π. (Since we are building a pool resolution refutation, not a w-resolution refutation, there is no need for $T_{i,j,k}$ to be an input lemma.)

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^{pool}. The guard variable thus is used as a resolution variable somewhere along the branch from the root to clause C. Then, as mentioned above, Lemma 25 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. 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 $C \subseteq C' \subseteq C^{\mathrm{pool}}$.

(iii) Suppose case (i) does not apply and that $x_{r,s}$ is not used as a resolution variable below $T_{i,j,k}$ in P_π and neither $x_{r,s}$ nor $\overline{x}_{r,s}$ is a member of C^{pool}. 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. It also means that 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_{t+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 only lemmas need to be repeated. The final line of S is the clause C', namely C plus the literals introduced by case (ii). The derivation R_{t+1} is formed from R_t by replacing the clause C with the derivation S of C', and then propagating each new literal $x \in C' \setminus C$ down towards the root of R_t, adding x to each clause below S until reaching a clause that already contains x. The derivation S contains no unfinished leaf, so R_{t+1} contains one fewer unfinished leaves than R_t.

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). In this case, 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 desired pool property. Hence, P_π is no longer used, and we instead will 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_{t+1} to be handled at later stages.

The construction of S is described in detail next, and depends on whether $T_{i,j,k}$ is type (β) or (γ). The base case of R_0 is type (β), but we describe the type (γ) construction first as it is somewhat simpler.

Suppose $T_{i,j,k}$ is type (γ), and thus $\overline{x}_{j,k}$ appears in C. (Refer to Figure 3.) Let $x_{r,s}$ be the guard variable for the transitivity axiom $T_{i,j,k}$. The derivation S will have the form

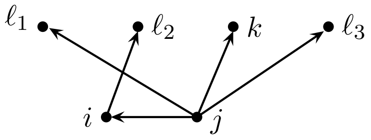
$$\dfrac{\dfrac{\dfrac{\overline{x}_{i,j},\overline{x}_{j,k},\overline{x}_{k,i},x_{r,s} \qquad \overline{x}_{i,j},\overline{x}_{j,k},\overline{x}_{k,i},\overline{x}_{r,s}}{\overline{x}_{i,j},\overline{x}_{j,k},\overline{x}_{k,i}} \qquad \begin{matrix} S_1 \\ \vdots \\ \overline{x}_{i,j},\overline{x}_{i,k},\overline{\pi}_{-[jk;jR(i)]}\end{matrix}}{\overline{x}_{i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

The notation $\overline{\pi}_{-[jk]}$ 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 $\overline{\pi}_{-[jk;iR(j)]}$ 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 $\overline{\pi}_{-[jk;jR(i)]}$ is defined similarly.

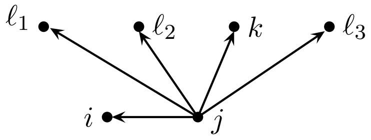
The upper leftmost inference of S is a resolution inference on the variable $x_{r,s}$. Since $T_{i,j,k}$ is not covered by either case (i) or (ii), the variable $x_{r,s}$ is not in C^{pool}. Thus, this use of $x_{r,s}$ as a resolution variable does not violate regularity. Furthermore, since $T_{i,j,k}$ 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 $x_{i,k}$ are not in C^{pool}, 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 dag, $\tau(C_2)$ and $\tau(C_1^-)$ are also dags. Let π_2 and π_1 be the bipartite orders associated with these dags (respectively). We will form the subderivation 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 $(\bigvee \overline{\pi}_2)$.

We first describe the subderivation S_2. The situation is pictured in Figure 5, which shows an extract from Figure 3: the edges shown in part (a) of the figure correspond to the literals present in the final line C_2 of S_2. In particular, recall that the 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 Figure 5.) The last line C_2 of S_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_{t+1}. (An example of this in Figure 5(a) is that



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



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

Figure 5: The partial orders for the fragment of S_2 shown in (7).

$j \prec_{\tau(C_2)} i \prec_{\tau(C_2)} \ell_2$, 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 Figure 5(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 $\overline{x}_{j,i}, \overline{\pi}_{-[jk;iR(j)]}$ from the clause $(\bigvee \overline{\pi}_2)$.

The subderivation of S_2 for replacing $\overline{x}_{i,\ell_2}$ in $\overline{\pi}$ with $\overline{x}_{j,\ell_2}$ in $\overline{\pi}_2$ is as follows, letting $\overline{\pi}^*$ be $\overline{\pi}_{-[jk;iR(j);i\ell_2]}$.

$$\dfrac{\begin{array}{c} S_2' \cdot \ddots \vdots \cdot \\ \overline{x}_{j,i}, \overline{x}_{i,\ell_2}, \overline{x}_{\ell_2,j} \end{array} \qquad \begin{array}{c} \ddots \vdots \cdot \text{ rest of } S_2 \\ \overline{x}_{j,k}, \overline{x}_{j,\ell_2}, \overline{x}_{j,i}, \overline{\pi}^* \end{array}}{\overline{x}_{j,k}, \overline{x}_{i,\ell_2}, \overline{x}_{j,i}, \overline{\pi}^*} \tag{7}$$

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 the transitivity axiom 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_t, 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_t 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 (7) 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 labeled with the clause $(\bigvee \overline{\pi}_2)$.

We next describe the subderivation S_1. The situation is shown in Figure 6. 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 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 Figure 6 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.

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

Figure 6: The partial orders for the fragment of S_1 shown in (8).

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 $\overline{\pi}_{-[jk;jR(i);j\ell_3]}$.

$$\dfrac{\overset{S_1' \;\vdots}{\overline{x}_{i,j}, \overline{x}_{j,\ell_3}, \overline{x}_{\ell_3,i}} \qquad \overset{\vdots\; \text{rest of } S_1}{\overline{x}_{i,k}, \overline{x}_{i,\ell_3}, \overline{x}_{i,j}, \overline{\pi}^*}}{\overline{x}_{i,k}, \overline{x}_{j,\ell_3}, \overline{x}_{i,j}, \overline{\pi}^*} \tag{8}$$

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 subderivation S_1' is formed in exactly 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 (8) 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 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 Figure 3.) In this case, S will have the form

$$\dfrac{\dfrac{\dfrac{\dfrac{T_{i,j,k}, x_{r,s} \qquad T_{i,j,k}, \overline{x}_{r,s}}{T_{i,j,k}} \qquad \overset{S_3 \;\vdots}{\overline{x}_{i,j}, \overline{x}_{i,k}, \overline{\pi}_{-[jR(i),kR(i\cup j)]}}}{\overline{x}_{i,j}, \overline{x}_{j,k}, \overline{\pi}_{-[jR(i),kR(i\cup j)]}} \qquad \overset{S_4 \;\vdots}{\overline{x}_{i,j}, \overline{x}_{k,j}, \overline{\pi}_{-[jR(i\cap k)]}}}{\overline{x}_{i,j}, \overline{\pi}_{-[jR(i\cap k)]}} \qquad \overset{S_5 \;\vdots}{\overline{x}_{j,i}, \overline{\pi}_{-[iR(j)]}}}{\overline{\pi}}$$

where $x_{r,s}$ is the guard variable for $T_{i,j,k}$. We write $[\overline{\pi}_{-[jR(i\cap k)]}]$ to mean the negations of literals in π omitting any literal $\overline{x}_{j,\ell}$ such that both $i \prec_\pi \ell$ and $k \prec_\pi \ell$. Similarly, $\overline{\pi}_{-[jR(i),kR(i\cup j)]}$ 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 either $i \prec_\pi \ell$ 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 subderivation S_5 is formed exactly like S_2 above, with the exception that now the literal $\overline{x}_{j,k}$ is not present. Thus we omit the description of S_5.

We next describe the construction of S_4. Let C_4 be the final clause of S_4; it is easy to check that $\tau(C_4)$ is a dag. As before, we must derive C_4 from the clause $(\bigvee \overline{\pi}_4)$ where π_4 is the bipartite partial order associated with $\tau(C_4)$. A typical situation is shown in Figure 7.

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

Figure 7: The partial orders as changed by S_4.

As pictured there, it is necessary to add the literals $\overline{x}_{i,\ell}$ such that $j \prec_\pi \ell$ and $i \not\prec_\pi \ell$, while removing $\overline{x}_{j,\ell}$; examples of this are ℓ equal to ℓ_2 and ℓ_3 in Figure 7. At the same time, we must add the literals $\overline{x}_{k,\ell}$ such that $j \prec_\pi \ell$ and $k \not\prec_\pi \ell$, while removing $\overline{x}_{j,\ell}$; examples of this are ℓ equal to ℓ_1 and, again, ℓ_2 in the same figure.

For a vertex ℓ_3 such that $j \prec_\pi \ell_3$ and $k \prec_\pi \ell_3$ but $i \not\prec_\pi \ell_3$, this is done similarly to the inferences (7) and (8) but without the side literal $\overline{x}_{j,k}$:

$$\dfrac{\begin{array}{c} S_4' \ \vdots \\ \overline{x}_{i,j}, \overline{x}_{j,\ell_3}, \overline{x}_{\ell_3,i} \end{array} \qquad \begin{array}{c} \vdots \ \text{rest of } S_4 \\ \overline{x}_{i,\ell_3}, \overline{x}_{k,j}, \overline{x}_{i,j}, \overline{\pi}^* \end{array}}{\overline{x}_{j,\ell_3}, \overline{x}_{k,j}, \overline{x}_{i,j}, \overline{\pi}^*} \tag{9}$$

Here $\overline{\pi}^*$ is $\overline{\pi}_{-[jR(i\cap k);j\ell_3]}$. The transitivity axiom T_{i,j,ℓ_3} shown as the last line of S_4' is handled exactly as before. This construction is repeated for all such ℓ_3's.

The vertices ℓ_1 such that $j \prec_\pi \ell_1$ and $i \prec_\pi \ell_1$ but $k \not\prec_\pi \ell_1$ are handled in exactly the same way. (The side literals π^* change each time to reflect the literals that have already been replaced.)

Finally, consider a vertex ℓ_2 such that $i \not\prec_\pi \ell_2$ and $j \prec_\pi \ell_2$ and $k \not\prec_\pi \ell_2$. This is handled by the derivation

$$\dfrac{\begin{array}{c} S_4'' \ \vdots \\ \overline{x}_{i,j}, \overline{x}_{j,\ell_2}, \overline{x}_{\ell_2,i} \end{array} \qquad \dfrac{\begin{array}{c} S_4''' \ \vdots \\ \overline{x}_{k,j}, \overline{x}_{j,\ell_2}, \overline{x}_{\ell_2,k} \end{array} \qquad \begin{array}{c} \vdots \ \text{rest of } S_4 \\ \overline{x}_{i,j}, \overline{x}_{i,\ell_2}, \overline{x}_{k,j}, \overline{x}_{k,\ell_2}, \overline{\pi}^* \end{array}}{\overline{x}_{i,j}, \overline{x}_{i,\ell_2}, \overline{x}_{k,j}, \overline{x}_{j,\ell_2}, \overline{\pi}^*}}{\overline{x}_{i,j}, \overline{x}_{k,j}, \overline{x}_{j,\ell_2}, \overline{\pi}^*}$$

As before, the set π^* of side literals is changed to reflect the literals that have already been added and removed as S_4 is being created. The subderivations S_4'' and S_4''' of the transitivity axioms T_{i,j,ℓ_2} and T_{k,j,ℓ_2} are handled exactly as before, depending on the status of their guard variables.

Finally, we describe how to form S_3. For this, we must form the bipartite partial order π_3 which is associated with $\tau(C_3)$, where C_3 is the final clause of S_3. To obtain $\overline{\pi}_3$, we need to add the literals $\overline{x}_{i,\ell}$ such that $i \not\prec_\pi \ell$ and such that either $j \prec_\pi \ell$ or $k \prec_\pi \ell$, while removing any literals $\overline{x}_{j,\ell}$ and $\overline{x}_{k,\ell}$. This is done by exactly the same construction used above in (9). The literals in $\overline{\pi}_{-[jR(i);kR(i\cup j)]}$ are exactly the literals needed to carry this out. The construction is quite similar to the above constructions, and we omit any further description.

That completes the description of how to construct the LR partial refutations R_t. The process stops once some R_t has no unfinished clauses. We claim that the process stops after polynomially many stages.

To prove this, recall that R_{t+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 for some bipartite order π. Since P_π has size $O(n^3)$, this means that the number of clauses in R_{t+1} is at most the number of clauses in R_t plus $O(n^3)$. Also, by construction, R_{t+1} has one fewer unfinished clauses than R_t. In case (*iv*) however, R_{t+1} is formed by adding up to $O(n)$ many clauses to R_t plus adding either two or three new unfinished leaf clauses. In addition, 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_t having no unfinished clauses for some $i \le 6\binom{n}{3} = O(n^3)$. Therefore there is a pool refutation of GGT_n with $O(n^6)$ lines. Since the GGT_n principle has $O(n^3)$ many clauses, the number of inferences in the refutation is bounded by a quadratic polynomial of the number of the clauses being refuted.

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 GT_n contain only clauses of size $O(n^2)$.
Q.E.D. Theorem 18 □

Theorem 19 is proved with nearly the same construction. In fact, the only change needed is the construction of S from P'_π. Recall that in the proof of Theorem 18, the pool derivation S was formed by using a depth-first traversal of P'_π. This is not sufficient for Theorem 19, since now the derivation S must use only input lemmas. Instead, we again use the same result as before of Buss et al. (2008, Thm. 3.3), which states that a (regular) dag-like resolution derivation can be transformed into a (regular) tree-like derivation with input lemmas. Forming S in this way from P'_π suffices for the proof of Theorem 19: the lemmas of S are either transitive closure axioms derived earlier in R_t or are derived by input subderivations earlier in the post-order of S. Since the transitive closure axioms that appeared earlier in R_t were derived by resolving two GGT_n axioms, the lemmas used in S are all input lemmas.

The transformation of the theorem of Buss et al. (2008, Thm. 3.3) may multiply the size of the derivation by the depth of the original derivation. The proofs P_π have depth $O(n)$, so the regRTI refutation has overall size $O(n^7)$. This completes the proof of Theorem 19.
□

5. Greedy and Unit Propagating CDCL

This section proves that the guarded graph tautology clauses GGT_n can be refuted by a polynomial-time greedy, unit propagating CDCL search without restarts. One difficulty is that we do not even know whether regular dag-like resolution refutations can be polynomially simulated by greedy, unit propagating CDCL without restarts. Therefore, we must first establish Theorems 26 and 30 showing that there are greedy, unit propagating CDCL refutations without restarts for the graph tautologies GT_n and $\mathrm{GT}_{\pi,n}$. Theorem 31 then gives the polynomial-time greedy, unit propagating CDCL algorithm (without restarts) for GGT_n. The intuition is that the CDCL search traverses the regRTI refutations for GGT_n

from Theorem 19, learning transitivity clauses $T_{i,j,k}$ whenever possible. We will also need to use the notion of "absorption" used by Atserias et al. (2011) and Pipatsrisawat and Darwiche (2011).

We give a quick overview of CDCL search algorithms and clause learning algorithms; for greater detail, see the works of Marques-Silva and Sakallah (1999) and Beame et al. (2004). Given a set of clauses Γ as input, the CDCL search maintains a stack of literals assigned the value *True* and a collection Δ of learned clauses. A simplified, but general, form of the procedure for CDCL without restarts is as follows:

Input: A set Γ of clauses.
Algorithm:
  Set $\Delta := \emptyset$.
  Loop:
    If unit propagation from $\Gamma \cup \Delta$ yields a contradiction,
      Halt. Γ is unsatisfiable.
    Else if unit propagation from the assigned literals and $\Gamma \cup \Delta$
          yields a contradiction,
      Infer zero or more clauses by input resolution based on the
          conflict, and add them to Δ. (Learning)
      Select $\Delta^- \subseteq \Delta$, and set $\Delta = \Delta \setminus \Delta^-$. (Garbage collection)
      Unassign the last assigned literal. (Backtracking)
    Else if all literals are assigned a value,
      Halt. Γ is satisfiable.
    Else
      Choose an unassigned literal, and assign it the value *True*.
    Endif

The above-described CDCL algorithm lacks many important features of the usual implementations of CDCL algorithms. However, this simplified CDCL algorithm can polynomially simulate more sophisticated implementations of CDCL by learning and unlearning the appropriate clauses. Conversely, more sophisticated CDCL algorithms can polynomially simulate the simplified CDCL algorithm, and since we are proving a polynomial time runtime for the simplified algoithm, this certainly implies a polynomial time runtime for more sophisticated CDCL algorithms (subject as usual to specifying which decision literals are set and which clauses are learned and garbage collected).

The clause learning used in Theorems 26 and 30 is not the usual first-UIP (unique implication point) clause learning, but does learn only clauses that are obtained by picking a cut in the conflict graph, as is common for CDCL algorithms.

Theorem 26. *Greedy, unit propagating CDCL search without restarts can refute the* GT_n *clauses in polynomial time.*

Proof. Recall the construction of the refutations P_n of the GT_n clauses as pictured in Figure 1. The key property needed is that the derivation of $\mathrm{Tot}_{k,j}$ from $\mathrm{Tot}_{k+1,k+1}$ and $\mathrm{Tot}_{k+1,j}$ is an input derivation, and this will permit learning $\mathrm{Tot}_{k,j}$ once $\mathrm{Tot}_{k+1,k+1}$ and $\mathrm{Tot}_{k+1,j}$ have been learned.

We will give a direct description of the CDCL search. The search initially chooses to set the literals $x_{1,0}$, $x_{2,1}\ldots$, $x_{n-2,n-3}$ true as decision literals, in that order, so $x_{i+1,i}$ is set true at (decision) level $i+1$. A literal $x_{i,j}$ is interpreted as meaning that $i \prec j$ in the partial order $\prec$; thus the decision literals express that $n-2 \prec n-3 \prec \cdots \prec 1 \prec 0$ holds. Unit propagation with the transitive closure clauses now implies the literals $x_{j,i}$, or in other words that $j \prec i$, for $0 \le i < j \le n-2$. The literal $x_{j,i}$ is derived at level j.

Until reaching the literal $x_{n-2,n-3}$, no other unit propagations are possible. But, upon assigning $x_{n-2,n-3}$ true, the clause $\mathrm{Tot}_{n-1,n-2}$ becomes a unit clause, and $x_{n-1,n-2}$ is inferred. In other words, $n-1 \prec n-2$ is inferred. Now, from the literals $x_{n-2,j}$, further unit propagations with transitive closure clauses gives $x_{n-1,j}$, that is $n \prec j$, for all $j < n-2$. This falsifies $\mathrm{Tot}_{n-1,n-1}$, and yields a conflict, so the CDCL search backtracks to the previous level.

As it backtracks, the CDCL algorithm learns the two clauses $\mathrm{Tot}_{n-3,n-3} \vee x_{n-3,n-2}$ and $\mathrm{Tot}_{n-2,n-2}$, and sets the decision literal $x_{n-2,n-3}$ false at level $n-3$. The first clause is learned by taking this decision literal $x_{n-2,n-3}$ as the UIP, since the literals $x_{n-3,j}$ for $j < n-3$ were the literals of level $< n-3$ that were used to infer level $n-2$ literals in the conflict graph. To see it is possible to learn the second clause $\mathrm{Tot}_{n-2,n-2}$, note that the conflict was obtained via unit propagation from the decision literal $x_{n-2,n-3}$ and the literals $x_{n-2,j}$, and that $\mathrm{Tot}_{n-2,n-2}$ contains exactly the negations of these literals. The literals $x_{n-2,j}$ were all set at level $n-2$, so $\mathrm{Tot}_{n-2,n-2}$ was not learned from a UIP; nonetheless, it falls within what is commonly permitted for CDCL learning. Once $\mathrm{Tot}_{n-3,n-3} \vee x_{n-3,n-2}$ is learned, the literal $x_{n-3,n-2}$ follows by a single unit propagation.

After this first backtrack, the decision literals $x_{1,0}$, $x_{2,1}, \ldots$, $x_{n-3,n-4}$ have been set true as decision literals, $\mathrm{Tot}_{n-2,n-2}$ and $\mathrm{Tot}_{n-3,n-3} \vee x_{n-3,n-2}$ are learned, and $x_{n-3,n-2}$ is true at level $n-3$ by unit propagation. It is possible to use unit propagation to get yet another conflict. However, instead of describing this, we go to the general cases.

In the general case, there are parameters ℓ, r, s so that $0 \le \ell < r < s < n$ with either $s = r+1$ or $s = n-1$. In this general case, the literals $x_{1,0}$, $x_{2,1}, \ldots$, $x_{\ell,\ell-1}$ are set true as the first ℓ decision literals and the literals $x_{\ell,\ell+2}$, $x_{\ell,\ell+3}, \ldots$, $x_{\ell,r}$ are set as the next decision literals, and the three clauses $\mathrm{Tot}_{s,\ell}$, $\mathrm{Tot}_{\ell,\ell} \vee x_{\ell,\ell+1}$, and $\mathrm{Tot}_{k,k}$ for all $k > \ell$ have been learned. (Refer to Figure 1. The situation in the previous paragraph has $\ell = n-3$, $r = n-2$, and $s = n-1$.) Using $\mathrm{Tot}_{\ell,\ell} \vee x_{\ell,\ell+1}$ and unit propagation, the literal $x_{\ell,\ell+1}$ is set at level ℓ. In terms of $\prec$, the literals which are set true express the conditions

$$\ell \prec \ell-1 \prec \cdots \prec 1 \prec 0 \quad \text{and} \quad \ell \prec j \text{ for } j = \ell+1, \ldots, r. \tag{10}$$

If $r < s-1 = n-2$, unit propagation does not yield a contradiction, and the CDCL search proceeds by setting $x_{\ell,r+1}$ true as the next decision literal. The CDCL search is now in the general case with parameters $\ell, r{+}1, n{-}1$.

When $r = s-1$, unit propagation with transitivity clauses yields $x_{\ell,j}$, namely $\ell \prec j$, for all $j \in [s] \setminus \{\ell\}$. This makes $\mathrm{Tot}_{s,\ell}$ a unit clause and thus $x_{s,\ell}$, namely $s \prec \ell$, is set true by unit propagation. Using the literals $x_{\ell,j}$ again, unit propagation with transitivity clauses falsifies the learned clause $\mathrm{Tot}_{s,s}$. The CDCL algorithm now backtracks one level. When $r > \ell+1$, it learns $\mathrm{Tot}_{s-1,\ell}$. This can be learned since the conflict was obtained from the literals $x_{\ell,j}$ for $j \in [s] \setminus \{\ell\}$. The learned clause $\mathrm{Tot}_{s,\ell}$ is forgotten by garbage collection. This puts the CDCL search in the new general situation with parameters $\ell, r{-}1, s{-}1$.

For $r = \ell + 1 = s - 1$, the two clauses $\mathrm{Tot}_{\ell,\ell}$ and $\mathrm{Tot}_{\ell-1,\ell-1} \vee x_{\ell-1,\ell}$ are learned instead. The second clause is the clause obtained by the usual learning algorithm using the decision literal $x_{\ell,\ell-1}$ as the UIP. The clause $\mathrm{Tot}_{\ell,\ell}$ can be learned since the conflict graph used the literals $x_{\ell,j}$ for $j < \ell$ to obtain the conflict: this is not obtained by a UIP, but it is obtained by by using a cut in the conflict graph. Garbage collection is used to forget the clauses $\mathrm{Tot}_{\ell,\ell} \vee x_{\ell,\ell+1}$ and $\mathrm{Tot}_{s,\ell}$. This puts the CDCL search in the general case, with the new ℓ, r, s parameters equal to $\ell - 1$, ℓ, and $n - 1$, respectively.

The CDCL search ends after dealing with the case $\ell = 0$, $r = 1$ and $s = 2$. In this case, the empty clause $\mathrm{Tot}_{0,0}$ is learned, and the CDCL search halts, establishing that GT_n is unsatisfiable. □

The next theorem discusses how a greedy, unit propagating CDCL search can simulate the derivations P_π of the graph tautologies $\mathrm{GT}_{\pi,n}$ obtained by restricting GT_n with a bipartite partial order π. Of course, this does not fully make sense, since CDCL search only simulates *refutations*, not *derivations*. What we really want is to simulate P_π as a subderivation of the regRTI refutations of GGT_n as constructed in Theorem 19. For this, let C be a clause in the regRTI refutation of GGT_n. Let τ be the dag such that $i \prec_\tau j$ iff $\overline{x}_{i,j} \in C^{\mathrm{pool}}$, and let π be the associated bipartite partial order, so C is the clause $(\bigvee \overline{\pi})$. We claim there is a greedy, unit propagating CDCL refutation of the (non-guarded) $\mathrm{GT}_{\pi,n}$ clauses from the literals $x_{i,j}$ such that $i \prec_\tau j$. In other words, greedy, unit propagating CDCL search can refute the GT_n clauses once it has set the literals in C^{pool} false.

To state Theorem 30 in full generality, we need the following definitions. These are modifications of definitions by Atserias et al. (2011) and Pipatsrisawat and Darwiche (2011).

Definition 27. Let C be a clause and x a literal in C. We say C is *u.p.-absorbed at x* by a set Γ of clauses provided that when every literal in $C \setminus \{x\}$ is set false, then x is implied by unit propagation using clauses from Γ. We say C is *u.p.-absorbed* by Γ provided it is u.p.-absorbed by Γ at every $x \in C$. We say C is *absorbed* by Γ provided that when all literals in C are set false, then unit propagation using Γ yields a contradiction.

Once a clause C becomes u.p.-absorbed, it becomes redundant in terms of unit propagation. In particular, adding C as an additional clause would not yield any additional conflicts and would not make it possible to learn any additional clauses from conflicts.

We need a slightly more general definition, however.

Definition 28. Let C and Γ be as above, and let L be a set of literals. We write $\overline{L}$ to denote the set of unit clauses $\{\overline{x}\}$ for $x \in L$, i.e. the clauses asserting every $x \in L$ is false. Then C is *L-absorbed* or *L-u.p.-absorbed (at x)* by Γ provided it is absorbed or u.p.-absorbed (at x) by $\Gamma \cup \overline{L}$, respectively.

Suppose C is L-u.p.-absorbed. The intuition is that Γ is the current set of initial and learned clauses for a CDCL search $\mathcal{S}$, and that the literals in L have all been set false (either as decision literals or by unit propagation). Then, unit propagation from $\Gamma \cup \overline{L}$ yields a contradiction if and only if unit propagation from $\Gamma \cup \overline{L} \cup \{C\}$ does. The learned clause for the former may need to include additional literals from L however. To state this formally:

Lemma 29. *Let Γ be a set of clauses, L be a set of literals set false as above, and C be L-u.p.-absorbed by Γ. Suppose unit propagation from $\Gamma \cup \overline{L} \cup \{C\}$ yields a contradiction and* *allows a clause D to be learned. Then unit propagation from $\Gamma \cup \overline{L}$ also yields a contradiction, and allows a clause D' to be learned such that $D' \subseteq D \cup L$.*

The proof of Lemma 29 is almost immediate from the definitions, and is left to the reader. Note that the additional literals from L which appear in D' are precisely the negations of the literals of $\overline{L}$ which are needed to simulate the invocation of C for unit propagation.

Theorem 30. *Let τ define a dag on $[n]$, and let $\pi \subseteq \tau$ be the bipartite partial order associated with τ. Then there is a greedy, unit propagating CDCL search that finds a refutation of the $\mathrm{GT}_{\pi,n}$ clauses and the unit clauses $x_{i,j}$ such that $i \prec_\tau j$.*

Proof. As in the proof of Lemma 25, let $M_\pi = [m]$ and J_k be such that $J_k \prec_\pi k$ for $k \geq m$. Let L be the set of literals $x_{i,j}$ such that $i \prec_\pi j$. Let $\mathcal{S}$ be the greedy, unit propagating CDCL search refuting the GT_m clauses of size polynomial in m given by Theorem 26. We wish to prove that $\mathcal{S}$ can be transformed into a greedy, unit propagating CDCL search $\mathcal{S}'$ that refutes the $\mathrm{GT}_{\pi,n}$ clauses and the clauses $\overline{L}$. The search $\mathcal{S}'$ will mimic $\mathcal{S}$ very closely, setting decision literals in the same order as $\mathcal{S}$, but will learn the clauses $\mathrm{Tot}^\pi_{i,j}$ in place of the clauses $\mathrm{Tot}_{i,j}$. (Compare Figures 1 and 4.)

$\mathcal{S}'$ must use the $\mathrm{GT}_{\pi,n}$ clauses as initial clauses instead of the GT_m clauses used by $\mathcal{S}$. First consider a GT_m transitivity clause used by $\mathcal{S}$; this is of type $(\beta_\emptyset)$ from Definition 17 and is equal to $T_{i,j,k}$ for distinct $i,j,k < m$. As such, it is also a $\mathrm{GT}_{\pi,n}$ clause of type (β) and thus is already available for $\mathcal{S}'$ to use.

Second, consider a totality clause $\mathrm{Tot}_{m-1,i} = \bigvee_{k \in [m]\setminus\{i\}} x_{k,i}$ of type $(\alpha_\emptyset)$ that is used for unit propagation by the search $\mathcal{S}$. This clause may not be L-u.p.-absorbed by the $\mathrm{GT}_{\pi,n}$ clauses. But, if not, then it is L-absorbed by $\mathrm{GT}_{\pi,n}$ and this will suffice for the search $\mathcal{S}'$ to succeed. To see that it is L-absorbed, suppose that $\mathcal{S}$ is at a point where unit propagation with $\mathrm{Tot}_{m-1,i}$ is used to obtain a conflict. In this case, all but one literal of $\mathrm{Tot}_{m-1,i}$ have been set false, namely there is a $j_0 \in [m] \setminus \{i\}$ so that $\mathcal{S}$ has set the literals $x_{j,i}$ for $j \in [m] \setminus \{i, j_0\}$ false and now infers $x_{j_0,i}$ by unit propagation. The new search $\mathcal{S}'$ also has set the literals $x_{j,i}$ for $j \in [m] \setminus \{i, j_0\}$ false and needs to infer $x_{j_0,i}$; but $\mathcal{S}'$ must use the $\mathrm{GT}_{\pi,n}$ clause $\mathrm{Tot}_{n-1,i}$ instead of the GT_m clause $\mathrm{Tot}_{m-1,i}$. Consider any $x_{k,i}$ in $\mathrm{Tot}_{n-1,i}$ with $k \geq m$. If there is a $j \in [m] \setminus \{i, j_0\}$ such that $j \prec_\pi k$ (we let J_k denote any such j), then, since $x_{j,k} \in L$, unit propagation with the transitivity clause $T_{i,j,k}$ sets $x_{i,k}$ true, i.e., falsifies $x_{k,i}$ in $\mathrm{Tot}_{n-1,i}$. If this holds for all $k \geq m$, then $\mathrm{Tot}_{n-1,i}$ is reduced to the unit $x_{j_0,i}$, so $\mathrm{Tot}_{m-1,i}$ is L-u.p.-absorbed at $x_{j_0,i}$ by $\mathrm{GT}_{\pi,n}$. Otherwise $\mathrm{Tot}_{m-1,i}$ is not L-u.p.-.absorbed. In this case, let $k_1, \ldots, k_R$ be the values $\geq m$ such that $j_0 \prec_\pi k_r$ for $1 \leq r \leq R$, so the just-described unit propagation is able to reduce $\mathrm{Tot}_{n-1,i}$ to the (non-unit) clause $x_{j_0,i} \vee x_{k_1,i} \vee \cdots \vee x_{k_R,i}$. The search $\mathcal{S}'$ now branches to set $x_{j_0,i}$ false as a decision literal. Unit propagation with the transitivity clauses T_{i,j_0,k_r}, for $r \leq R$ falsifies the literals $x_{k_r,i}$ and thereby falsifies $\mathrm{Tot}_{n-1,i}$. Then $\mathcal{S}'$ backtracks and learns the clause $S^\pi_i \vee \mathrm{Tot}_{m-1,i}$ (see (5)). This of course is just the clause $\mathrm{Tot}^\pi_{m-1,i}$. With $\mathrm{Tot}^\pi_{m-1,i}$ learned, and since the literals in S^π_i are in L and are set false, $\mathcal{S}'$ can now obtain a conflict in the same as way as $\mathcal{S}$.

In general, the literals in $\mathrm{Tot}^\pi_{i,j} \setminus \mathrm{Tot}_{i,j}$ are all members of L, and so are set false by $\mathcal{S}'$. It is thus straightforward to check that the remaining steps of $\mathcal{S}$ can be simulated directly by $\mathcal{S}'$ using exactly the same decision literals, and obtaining conflict clauses in the same way, but now learning the clauses $\mathrm{Tot}^\pi_{i,j}$ instead of the clauses $\mathrm{Tot}_{i,j}$. ☐

We can now describe the greedy and unit propagating CDCL without restart procedures that refute the GGT_n clauses.

Theorem 31. *There are CDCL without restart search procedures which are greedy and unit propagating, and refute the* GGT_n *clauses in polynomial time.*

Proof. The basic idea for the greedy, unit propagating CDCL search $\mathcal{S}$ is that it follows the structure of the refutation described in the proofs of Theorems 18 and 19; that is, $\mathcal{S}$ follows the stages of the construction of LR-partial refutations.

At a stage of $\mathcal{S}$ that corresponds to an unfinished clause C of a LR-partial refutation, S has some set of transitivity clauses $T_{i,j,k}$ as its only learned clauses, and has set zero or more literals $x_{i,j}$ true. These literals describe a dag τ, namely $i \prec_\tau j$ iff $x_{i,j}$ has been set true. Let π be the associated bipartite partial order, so $C = (\bigvee \overline{\pi})$. Then S has set every $x_{i,j}$ true for which $i \prec_\pi j$. If possible, S will now use the refutation of Theorem 30 to backtrack from the current stage; otherwise, S will branch to learn a another transitivity clause. The first possibility holds provided that every transitivity clause $T_{i,j,k}$, of type (β) or (γ) needed for the refutation of $\mathrm{GT}_{\pi,n}$ is either learned or C^{pool}-u.p.-absorbed by S.

Otherwise, suppose some transitivity clause $T_{i,j,k}$ of type (γ) is neither learned nor C^{pool}-u.p.-absorbed. Following the pattern of the argument for this case in the proof of Theorem 18, S sets $x_{i,j}$ true as a decision literal. Since $T_{i,j,k}$ was of type (γ), $x_{i,j}$ has not already been set. We claim that setting $x_{i,j}$ true does not yield a contradiction by unit propagation. To prove this claim, note that the only way unit propagation can occur is if some (guarded or unguarded) transitivity clause becomes a unit clause. Referring back to Figure 3, the only (possibly guarded) transitivity clauses which can become unit upon setting $x_{i,j}$ true are clauses $T_{i,j,\ell}$ for $\ell \geq m$ such that $\mathcal{S}$ has set $x_{j,\ell}$ true. Therefore unit propagation does not yield a contradiction from $x_{i,j}$. S now sets $x_{k,i}$ true. Since $T_{i,j,k}$ is of type (γ) and $C = (\bigvee \overline{\pi})$, S has set $x_{j,k}$ true; therefore, unit propagation yields a contradiction from $T_{i,j,k} \vee \rho$ and $T_{i,j,k} \vee \overline{\rho}$. From this, S backtracks, setting $x_{i,k}$ true and learning $T_{i,j,k}$.

After backtracking, S is at a stage corresponding to the clause C_1 of the subrefutation S_1 in the proof of Theorem 18. Let L_1 be the literals that are set false, τ_1 be the dag defined by the literals of L_1, and let π_1 be the associated bipartite order. $\mathcal{S}$ needs to infer the literals $x_{i,\ell}$ for $i \prec_{\pi_1} \ell$. This is done using transitivity clauses. Any L_1-u.p.-absorbed transitivity clause can be used directly. For other transitivity clauses $T_{i,j,\ell}$, once $x_{i,j}$ and $x_{j,\ell}$ are set true, then S branches to set $x_{\ell,i}$ false as a decision literal, obtains an immediate contradiction by unit propagation from the two GGT_n clauses for $T_{i,j,\ell}$ and backtracks, learning $T_{i,j,\ell}$ and inferring $x_{i,\ell}$ by unit propagation. Once all the literals $x_{i,\ell}$ where $i \prec_{\pi_1} \ell$ are set true, S is again at a stage corresponding to an unfinished clause.

Once S backtracks out of the stage corresponding to the clause C_1, S enters the stage corresponding to C_2. This is handled similarly.

The argument for the case where a transitivity clause $T_{i,j,k}$ of type (β) has not been learned is very similar. Again we follow the construction in the proof of Theorem 18, and now use stages that correspond to clauses C_3, C_4, and C_5. Since the construction is quite similar, we omit its description. ∎

Theorem 31 shows a very close correspondence between the regRTI proofs of the GGT_n clauses and greedy, unit propagating CDCL refutations without restarts. It is open, however, whether an arbitrary regRTI or regWRTI refutation can be converted into a polynomial size greedy, unit propagating CDCL refutation without restarts. It is even open whether every (dag-like) regular refutation corresponds to a greedy, unit propagating CDCL refutation without restarts.

Acknowledgements

We are grateful to J. Hoffmann for assisting with a correction to an earlier version of the proof of Theorem 19. We also thank A. Van Gelder, L. Kołodziejczyk, A. Beckmann, T. Pitassi, and three anonymous referees for encouragement, suggestions, and useful substantial comments.

M. L. Bonet was supported in part by grant TIN2010-20967-C04-02, and by a Research Abroad Fellowship (Generalitat de Catalunya BE, 2012).

S. Buss was supported in part by National Science Foundation grants DMS-0700533, DMS-1101228 and CCF-1213151, and by a grant from the Simons Foundation (#208717 to Sam Buss). He also thanks the John Templeton Foundation for supporting his participation in the CRM Infinity Project at the Centre de Recerca Matemàtica, Barcelona, Catalonia, Spain during which some of these results were obtained.

S. Buss and J. Johannsen thank the Banff International Research Station for the workshop on Proof Complexity (11w5103) held in October 2011 during which part of these results were obtained.

References

Alekhnovich, M., Johannsen, J., Pitassi, T., & Urquhart, A. (2007). An exponential separation between regular and general resolution. *Theory of Computing*, *3*(5), 81–102.

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

Atserias, A., Fichte, J. K., & Thurley, M. (2011). Clause-learning algorithms with many restarts and bounded-width resolution. *Journal of Artificial Intelligence Research*, *40*, 353–373.

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

Beckmann, A., & Buss, S. R. (2005). Separation results for the size of constant-depth propositional proofs. *Annals of Pure and Applied Logic*, *136*, 30–55.

Ben-Sasson, E. (2009). Size space tradeoffs for resolution. *SIAM Journal on Computing*, *38*(6), 2511–2525.

Ben-Sasson, E., Impagliazzo, R., & Wigderson, A. (2004). Near optimal separation of tree-like and general resolution. *Combinatorica*, *24*(4), 585–603.

Bonet, M. L., & Buss, S. R. (2012a). An improved separation of regular resolution from pool resolution and clause learning. In *Proc. 15th International Conference on Theory and*

Applications of Satisfiability Testing – SAT 2012, Lecture Notes in Computer Science #7317, pp. 45–57.

Bonet, M. L., & Buss, S. R. (2012b). An improved separation of regular resolution from pool resolution and clause learning. Full version, arxiv.org, arXiv:1202.2296v2 [cs.LO].

Bonet, M. L., & Galesi, N. (2001). Optimality of size-width tradeoffs for resolution. *Computational Complexity*, *10*(4), 461–474.

Buss, S., & Kołodziejczyk, L. (2012). Small stone in pool. Submitted for publication.

Buss, S. R. (2009). Pool resolution is NP-hard to recognise. *Archive for Mathematical Logic*, *48*(8), 793–798.

Buss, S. R., Hoffmann, J., & Johannsen, J. (2008). Resolution trees with lemmas: Resolution refinements that characterize DLL-algorithms with clause learning. *Logical Methods in Computer Science*, *4, 4:13*(4:13), 1–18.

Chang, C. L. (1970). The unit proof and the input proof in theorem proving. *J. ACM*, *17*(4), 698–707.

Goerdt, A. (1993). Regular resolution versus unrestricted resolution. *SIAM Journal on Computing*, *22*(4), 661–683.

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

Huang, W., & Yu, X. (1987). A DNF without regular shortest consensus path. *SIAM Journal on Computing*, *16*(5), 836–840.

Johannsen, J. (2009). An exponential lower bound for width-restricted clause learning. In *Proc. 12th International Conference on Theory and Applications of Satisfiability Testing – SAT 2009*, Lecture Notes in Computer Science #5584, pp. 128–140.

Krishnamurthy, B. (1985). Short proofs for tricky formulas. *Acta Informatica*, *22*(3), 253–275.

Marques-Silva, J. P., & Sakallah, K. A. (1999). GRASP — A new search algorithm for satisfiability. *IEEE Transactions on Computers*, *48*(5), 506–521.

Pipatsrisawat, K., & Darwiche, A. (2011). On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, *172*(2), 512–525.

Segerlind, N., Buss, S. R., & Impagliazzo, R. (2004). A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM Journal on Computing*, *33*(5), 1171–1200.

Stålmarck, G. (1996). Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, *33*(3), 277–280.

Urquhart, A. (2011). A near-optimal separation of regular and general resolution. *SIAM Journal on Computing*, *40*(1), 107–121.

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

Van Gelder, A. (2006). Preliminary report on input cover number as a metric for propositional resolution proofs. In *Theory and Applications of Satisfiability Testing - SAT 2006*, Lecture Notes in Computer Science 4121, pp. 48–53. Springer Verlag.