SMALL STONE IN POOL

SAMUEL R. BUSS\textsuperscript{a} AND LESZEK ALEKSANDER KOŁODZIEJCZYK\textsuperscript{b}

\textsuperscript{a} Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112, USA
e-mail address: sbuss@math.ucsd.edu

\textsuperscript{b} Institute of Mathematics, University of Warsaw, Banacha 2, 02-097 Warszawa, Poland
e-mail address: lak@mimuw.edu.pl

Abstract. The Stone tautologies are known to have polynomial size resolution refutations and require exponential size regular refutations. We prove that the Stone tautologies also have polynomial size proofs in both pool resolution and the proof system of regular tree-like resolution with input lemmas (regRTI). Therefore, the Stone tautologies do not separate resolution from DPLL with clause learning.

\textit{I have said it thrice; What I tell you three times is true.}
\textit{Fit the First — The Landing; The Hunting of the Snark}
Lewis Carroll

1. Introduction

The Davis-Putnam-Logemann-Loveland (DPLL) proof search method \cite{DLL62,DP60}, augmented with clause learning \cite{MSS99}, has become a core method for solving the satisfiability (SAT) problem, especially for large-scale instances of SAT that arise in industrial applications. However, when restarts are not allowed, the proof strength of DPLL with clause learning relative to full resolution remains unknown. On one hand, if $\Gamma$ is a set of clauses, and DPLL with clause learning can show that $\Gamma$ is unsatisfiable in $n$ steps, then $\Gamma$ has a resolution refutation with size polynomially bounded by $n$ (see \cite{BKS04}). On the other hand, the results of \cite{AJPU07,Urq11,BB12,BBJ14} imply that the length of DPLL with clause learning proof searches can be nearly exponentially smaller than the length of the shortest regular resolution proofs. Systems designed to correspond to DPLL with clause learning, such as pool resolution (\cite{VC05}) and regRTI (\cite{BHJ14}), are therefore simulated.
by resolution and strictly stronger than regular resolution. Determining the exact strength of those systems is an open problem.

The two papers [AJPU07, Urq11] gave examples of three principles which have polynomial size resolution refutations, but require exponential size regular refutations. In the terminology of [BB12, BBJ14], these three principles were (1) the guarded graph tautologies, (2) the Stone tautologies, and (3) the guarded pebbling tautologies. Subsequently, [BB12, BBJ14] showed that the guarded graph tautologies and the guarded pebbling tautologies have polynomial size pool and regRTI refutations, and hence can be refuted by polynomially long DPLL search with clause learning and without restarts.

It remained open whether the same holds for the Stone tautologies. There seems to be an inherent simplicity in the irregularity introduced by “guarded” versions of combinatorial principles, such as (1) and (3). This is because the guarded principles have refutations in which all irregularities are at the initial inferences; namely, the resolution refutation can start by using resolution to remove the guard literals, and then give a (regular) refutation of the underlying principle as usual. In contrast, the prior known resolution refutations for the Stone principles of [AJPU07] use irregularity in a more essential fashion, with the irregularities distributed throughout the refutation. Because of this, it was conjectured that the Stone principles might be examples where the pool and regRTI systems, and thus DPLL with clause learning and no restarts, require exponential size refutations. That is, the Stone tautologies were viewed as candidates for separating DPLL with clause learning from resolution.

The present paper, however, refutes this conjecture and establishes that the Stone tautologies do indeed have polynomial size pool refutations and regRTI refutations. In light of this, the possibility that pool and regRTI actually simulate the full power of resolution perhaps becomes slightly more plausible. Nevertheless, even if such a general simulation result does hold, it is far from clear whether the methods we use to deal with the Stone tautologies can be of much help in proving it.

The remainder of this introduction gives a review of the basic definitions, first of the Stone principles, then of the various proof systems. It concludes by stating our main theorems about the existence of pool and regRTI refutations. The reader is encouraged to consult the introductory sections of [BB12, BBJ14] for a more extensive discussion of prior work, and to consult [AJPU07] for more on the Stone principles. A good general introduction to DPLL with clause learning is [BKS04].

**Definition 1.1.** A **literal** is either a propositional variable $x$ or a negated variable $\overline{x}$. A **clause** is a set of literals, usually written as a list of literals separated by either $\lor$’s (disjunctions) or commas. A clause is interpreted as the disjunction of its members. A set $\Gamma$ of clauses is interpreted as the conjunction of its members, so $\Gamma$ represents a propositional formula in conjunctive normal form.

The next definition describes the Stone principle of [AJPU07] as a set of clauses. The Stone principle is a kind of induction principle. For a given directed acyclic graph (dag), it states that if each source vertex is pebbled with a red stone and if each vertex whose immediate predecessors are pebbled with red stones is also pebbled with a red stone, then the sink vertex is pebbled with a red stone.

**Definition 1.2.** Assume that $G = (V, E)$ is a dag with a single sink, with vertices $V = \{1, \ldots, N\}$, such that each non-source vertex of $G$ has in-degree 2. We assume that vertices are numbered consistently with the directions of the edges of $G$ so that if there is an edge
We next recall the definitions of various types of resolution. The literal \(x\) is the \textit{resolution variable}. Three kinds of inferences are defined by:

\begin{itemize}
  \item \textbf{Resolution rule:} We have \(x \in A\), \(\overline{x} \in B\), and \(C = (A \setminus \{x\}) \lor (B \setminus \{\overline{x}\})\).
  \item \textbf{Degenerate resolution rule:} \[\text{If } x \in A \text{ and } \overline{x} \in B, \text{ then } C \text{ is obtained as in the resolution rule. If } x \in A \text{ and } \overline{x} \notin B, \text{ then } C \text{ is } B. \text{ If } x \notin A \text{ and } \overline{x} \in B, \text{ then } C \text{ is } A. \text{ Otherwise } C \text{ is one of } A \text{ or } B.\]
  \item \textbf{w-resolution rule:} \[\text{The clause } C \text{ equals } (A \setminus \{x\}) \lor (B \setminus \{\overline{x}\}).\]
\end{itemize}

The three different types of resolution coincide when \(x \in A\) and \(\overline{x} \in B\), in which case we refer to the inference as \textit{non-degenerate}.

\textbf{Definition 1.4.} A \textit{resolution derivation} \(D\) of a clause \(C\) from a set \(\Gamma\) of clauses is a sequence of clauses \(C_1, \ldots, C_s = C\) and such that each \(C_i\) is either a clause from \(\Gamma\) or is inferred by a resolution rule from two previous clauses. If \(C\) is the empty clause, \(D\) is a \textit{resolution refutation} of \(\Gamma\). \textit{Degenerate resolution} and \textit{w-resolution} derivations and refutations are defined similarly.

The \textit{size} of a refutation \(C_1, \ldots, C_s = \bot\) is defined to be \(s\).

Derivations are typically viewed as directed acyclic graphs. A derivation is \textit{tree-like} provided its dag is a tree. It is well known that (tree-like) resolution is sound and complete, in that \(\Gamma\) has a refutation iff it is unsatisfiable.
Definition 1.5. A refutation \( D \) is regular provided that no variable is used as a resolution variable more than once along any path in the directed acyclic graph of \( D \). A derivation \( D \) of a clause \( C \) is regular provided that, in addition, no variable appearing in \( C \) is used as a resolution variable in \( D \).

We next define “regular resolution derivation trees with lemmas”, or “regRTL”, following [BHJ08]. The idea is that a dag-like proof can be rewritten as a tree-like proof in which clauses obtained earlier in the proof can be used freely as “learned” lemmas. This will be the key component in defining Van Gelder’s notion of pool proofs.

Definition 1.6. Given a tree \( T \), the postorder ordering \( <_T \) of the nodes is defined as follows: if \( u, v, w \) are distinct nodes 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 \( v <_T w <_T u \). The preorder ordering \( <'_T \) is defined similarly, but stipulates that \( u <'_T v <'_T w \).

Definition 1.7. A regRTL derivation [BHJ08] of a clause \( C \) from a set of initial clauses \( \Gamma \) is a tree-like resolution derivation \( T \) that fulfills the following conditions: (a) each leaf is labeled with either a clause of \( \Gamma \) or a clause (called a “lemma”) that appears earlier in \( T \) 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 \( C \). If the labeling of the root is the empty clause, \( T \) is a regRTL refutation.

A regWRTL derivation [BHJ08] is defined similarly, but allowing \( w \)-resolution inferences instead of just resolution inferences.

A pool resolution derivation [VG05] is also defined similarly, but allowing degenerate resolution inferences.

Proposition 1.8. If \( \Gamma \) has a regWRTL refutation \( R \), then \( \Gamma \) has a pool resolution refutation \( R' \) with the size of \( R' \) no greater than the size of \( R \).

The proof of Proposition 1.8 is simple. Each clause \( C \) in \( R \) corresponds to a clause \( C' \) in \( R' \) with \( C' \subseteq C \). Arguing inductively, suppose that \( C \) is derived in \( R \) from the clauses \( C_1 \) and \( C_2 \) using resolution literal \( x \). Then, it is straightforward to define \( C' \) from \( C'_1 \) and \( C'_2 \) as the unique clause that can be inferred by degenerate resolution from \( C'_1 \) and \( C'_2 \) with respect to \( x \).

The strategy of proving the existence of short pool refutations via constructing short regWRTL refutations is employed in the proof of Theorem 1.14 below.

Definition 1.9. ([BHJ08]). A “lemma” in clause (a) of the definition of regRTL derivations is called an input lemma if it is derived by an input subderivation, namely by a subderivation in which each inference has at least one hypothesis which is a member of \( \Gamma \) or a lemma. A regRTI derivation is a regRTL derivation which uses only input lemmas as lemmas.

A bit more generally, we say that a clause is “learned” provided it is available for use as a lemma by virtue of having been learned earlier in the postorder traversal of the proof, or by virtue of being an initial clause:

Definition 1.10. Suppose that \( R \) is a regRTL (respectively, a regRTI) refutation of \( \Gamma \), and let \( C \) be a clause in \( R \). The learned clauses of \( R \) at clause \( C \) are the clauses which are either in \( \Gamma \) or which have been derived in \( R \) (respectively, have been derived by an input subderivation in \( R \)) before \( C \) in the postordering of \( R \).
Theorem 5.1 of [BHJ08] gives a polynomial equivalence between regRTI proofs and DPLL with clause learning without restarts. This equivalence, however, uses non-greedy DPLL; namely, the DPLL proof search may need to ignore contradictions during its search. Since most real-world DPLL search algorithms do not ignore contradictions, and use unit propagation whenever possible, is it natural to posit similar properties for regRTI proofs. These are formalized by the next two definitions.

**Definition 1.11.** Let $C$ be a clause appearing in a regular derivation $R$. Following [BBJ14], we write $C^\text{pool}$ to denote the clause containing the literals that appear in any clause in the path from the root of $R$ up to and including $C$.

The clause $C^\text{pool}$ is the same as what [BB12] calls $C^+$. The regularity of $R$ ensures that $C^\text{pool}$ contains no contradictory literals.

**Definition 1.12.** Let $R$ be a tree-like refutation of $\Gamma$. A clause $D$ in $R$ is prior-learned for a clause $C$ in $R$ if either $D \in \Gamma$ or there is an occurrence of $D$ as a learned clause which appears in $R$ before $C$ in both postorder and preorder.

The intuition for “prior-learned”, is that, when reaching the clause $C$ while constructing $R$ in left-to-right, depth-first order, the prior-learned clauses are the clauses that are already available to help derive $C$.

**Definition 1.13.** (See [BB12]). A refutation $R$ is greedy provided that, for each clause $C$ of $R$, if $C$ or any subclause of $C$ is prior-learned, then $C$ itself is a prior-learned clause and is a leaf clause of $R$. A refutation $R$ is greedy and unit-propagating provided that, for each clause $C$ of $R$, if there is an input derivation of some clause $C' \subseteq C^\text{pool}$ from the prior-learned clauses of $R$ at $C$ which does not resolve on any literal in $C^\text{pool}$, then $C$ is derived in $R$ by such a derivation.

We can now state our main results.

**Theorem 1.14.** The Stone principles $\text{Stone}(G, m)$ have regWRTL refutations, and thus pool refutations, of size $O(Nm^3)$.

**Theorem 1.15.** The Stone principles $\text{Stone}(G, m)$ have regRTI refutations of size $O(N^3m^4)$.

It follows from Theorem 1.15 and Theorem 5.1 of [BHJ08] that DPLL proof search with clause learning and without restarts can refute the Stone principle clauses in polynomial time. It is possible that the regRTI refutations of Theorem 1.15 can be made greedy and unit-propagating, but we have not tried to prove this.

The proofs of Theorems 1.14 and 1.15 are given in Sections 3 and 4 respectively. Section 2 first gives some preliminary resolution derivations that will be useful for both proofs.

Of course, Theorem 1.15 implies Theorem 1.14 apart from the size bounds. However, it seems useful to prove the two theorems separately, since the proof of Theorem 1.14 is substantially simpler than the proof of Theorem 1.15.

The intuition behind both proofs is similar. The reason the Stone tautologies seem highly irregular is that, in the earlier refutations given by [AJPU07], some of the variables (the $r_j$'s) are resolved on repeatedly during the refutation. The intuition is that the regWRTL/regRTI proofs for Theorems 1.14 and 1.15 can be built in a bottom-up fashion starting from the empty clause, by first resolving on the variables $p_{i,j}$ that do not cause irregularities, and saving the problematic variables $r_j$ to be resolved on later (higher in the
proof). This is not quite completely true, since our derivations do also resolve again on $p_{i,j}$'s at the top of the derivations; it is nonetheless a useful intuition.

2. Learning and 3-Learning

The regWRTL refutation for Theorem 1.13 and the regRTI refutation for Theorem 1.15 both work by learning the clauses $\overline{p}_{i,j}, r_j$. If $i$ is a source vertex of $G$ then these clauses are Stone clauses, but otherwise they must be learned.

Suppose that $i$ is a non-leaf vertex, and $i'$ and $i''$ are the two predecessors of $i$ in $G$. In addition, suppose that every clause $\overline{p}_{i',j}, r_j$ and $\overline{p}_{i'',j}, r_j$ has already been learned. Fix a value of $j$. A derivation of $\overline{p}_{i,j}, r_j$ proceeds in the following three steps.

First, for each $j' \neq j''$, both distinct from $j$, derive the clause

$$\overline{p}_{i',j}', \overline{p}_{i'',j''}, \overline{p}, r_j \quad (2.1)$$

by resolving a Stone clause against the two learned clauses $\overline{p}_{i',j'}, r_{j'}$ and $\overline{p}_{i'',j''}, r_{j''}$ using $r_{j'}$ and $r_{j''}$ as resolution variables:

$$\overline{p}_{i',j'}, \overline{p}_{i',j''}, \overline{p}, r_j \quad \overline{p}_{i',j'}, r_{j'}, \overline{p}_{i'',j''}, \overline{p}, r_j \quad \overline{p}_{i'',j''}, r_{j''}$$

For $j' = j'' \neq j$, the clause (2.1) is derived in one step by resolving the Stone clause against only one of the two learned clauses $\overline{p}_{i',j'}, r_{j'}$ and $\overline{p}_{i'',j''}, r_{j''}$.

Second, for each $j'' \neq j$, resolve the Stone clause $\bigvee_{j' = 1}^{m} p_{i',j'}$ against the learned clause $\overline{p}_{i,j}, r_j$ and against $m - 1$ of the clauses (2.1) to obtain

$$\overline{p}_{i'',j''}, \overline{p}_{i,j}, r_j \quad (2.2)$$

This is shown in Figure 1.

Third, resolve the Stone clause $\bigvee_{j' = 1}^{m} p_{i'',j''}$ against the learned clause $\overline{p}_{i'',j''}, r_j$ and against the $m - 1$ many clauses (2.2), and derive the desired clause $\overline{p}_{i,j}, r_j$. This is shown in Figure 2.

For the regWRTL proof constructed in Section 3, the clause $\overline{p}_{i,j}, r_j$ will be learned and available to use as a lemma once the above three steps have been carried out.

For the regRTI proof described in Section 4, this is not sufficient, since only clauses derived by input subderivations are learned. For regRTI proofs, the above three steps are used the first time $\overline{p}_{i,j}, r_j$ is derived. This results in the clauses (2.1) being learned as input lemmas, but not the clauses (2.2). The second time $\overline{p}_{i,j}, r_j$ is derived, only the second and third steps of the above derivation are carried out. This results in the clauses (2.2) becoming learned, but not the clause $\overline{p}_{i,j}, r_j$. The third time $\overline{p}_{i,j}, r_j$ is derived, only the third step is needed; this results in the clause $\overline{p}_{i,j}, r_j$ becoming learned as an input lemma.

This leads to the following definition, which will be useful for the regRTI derivations of Section 4.

**Definition 2.1.** Let $i, i', i'', j$ be as above, so in particular all the clauses $\overline{p}_{i',j'}, r_{j'}$ and $\overline{p}_{i'',j''}, r_{j''}$ are learned. The clause $\overline{p}_{i,j}, r_j$ is called 3-learned provided it has been learned. It is called 2-learned if all of the clauses (2.2) for $j'' \neq j$ have been learned. It is called 1-learned if all of the clauses (2.1) for $j' \neq j, j'' \neq j$ have been learned.
Figure 1: The derivation of a clause (2.2) follows this pattern with the one exception (not shown) that the clause $p_{i,j}$ is not one of the $m - 1$ clauses (2.1) and the learned clause $\overline{p}_{i,j}$ is used instead.

Figure 2: The derivation of the clause $\overline{p}_{i,j}$ follows this pattern with the one exception (not shown) that the clause $\overline{p}_{i,j}$ is not one of the $m - 1$ clauses (2.2) and the clause $\overline{p}_{i,j}$ is used instead.

A vertex $i$ is defined to be $K$-learned, for $K = 1, 2, 3$, if and only if every $\overline{p}_{i,j}$ has been $K$-learned. It is also allowed that $K = 0$: every clause $\overline{p}_{i,j}$ and every vertex $i$ is considered to be 0-learned.
Since axiom clauses are considered to be learned, the source vertices \( i > n \) are 3-learned by definition.

The next theorem summarizes the above construction.

**Theorem 2.2.** Let \( i' \) and \( i'' \) be the two predecessors of vertex \( i \). There is a regular tree-like derivation of the clause \( \overline{p}_{i,j}^{'} r_j \) from Stone clauses and the clauses \( \overline{p}_{i'',j''}^{'} r_{j''} \) and \( \overline{p}_{i'',j''}^{''} r_{j''} \), which has size \( O(m^2) \) and resolves on (only) the variables \( r_k \) for \( k \neq j \) and the variables \( p_{r,k}^{'} \) and \( p_{r,k}^{''} \) for all \( k \).

In the setting of a regRTI proof, if \( i' \) and \( i'' \) are 3-learned, and \( \overline{p}_{i,j}^{'} r_j \) was already K-learned for \( K < 3 \), then there is a regular tree-like derivation of \( \overline{p}_{i,j}^{'} r_j \) from learned clauses (including Stone clauses) which causes it to become \( (K+1) \)-learned. This derivation has size \( O(m^2) \) and resolves on at most the variables \( r_k \) for \( k \neq j \) and variables \( p_{r,k}^{'} \) and \( p_{r,k}^{''} \) for all \( k \).

It will also be useful to modify the derivations described above to allow side variables \( \overline{p}_{j_1}, \ldots, \overline{p}_{j_l} \). This is summarized by the next theorem.

**Theorem 2.3.** Let \( i, i', i'' \) be as above. Let \( F = \{ \overline{p}_{j_1}, \ldots, \overline{p}_{j_k} \} \) where \( \overline{p}_{j} \notin F \). There is a regular tree-like derivation of the clause \( F, \overline{p}_{i,j}^{'} r_j \) from Stone clauses and the clauses \( \overline{p}_{i'',j''}^{'} r_{j''} \) and \( \overline{p}_{i'',j''}^{''} r_{j''} \), which has size \( O(m^2) \) and resolves on (only) the variables \( r_k \) for \( k \notin \{j, j_1, \ldots, j_k\} \) and the variables \( p_{r,k}^{'} \) and \( p_{r,k}^{''} \) for all \( k \).

The derivation for Theorem 2.3 is obtained from the derivation for Theorem 2.2 by omitting inferences that resolve on the literals \( r_{j_q} \) against the clauses \( \overline{p}_{i',j_q}^{'} r_{j_q} \) and \( \overline{p}_{i'',j_q}^{''} r_{j_q} \).

### 3. The pool/regWRTL refutation

This section proves Theorem 1.14 by describing regWRTL proofs of the Stone principles. Fix an instance of the Stone principle for a dag \( G \) as above with \( N \) vertices and \( m \) stones. Recall that \( G \) has \( n < N \) non-source vertices.

The regWRTL refutation of \( \text{Stone}(G,m) \) will be a tree with its final, empty, clause at the bottom. The main part of the regWRTL refutation above the empty clause is a “skeleton”, which consists of a long branch containing \( n \) segments of length \( m \) each, as is shown in Figures 5 and 6. Each segment in the skeleton corresponds to a non-source vertex in \( G \), and the role of the \( i \)-th segment is that clauses of the form \( \overline{p}_{i,j}^{'} r_j \) are learned on branches to the right of the segment. In keeping with the intuition discussed in the introduction, this skeleton is the bottom part of the proof which resolves on the literals \( p_{i,j} \); the variables \( r_j \) (plus additional variables \( p_{r,j} \) with \( i' > i \)) will be resolved on above the skeleton.

The first, second, and last segments of the skeleton are pictured in Figure 5. Each of these three segments is somewhat atypical, but the \( i \)-th segment for a typical intermediate \( i \in \{3, \ldots, n-1\} \) is pictured in Figure 4. In the typical situation, the idea is that for given \( 3 \leq i \leq n-1 \) and \( j < m \), the clause \( \overline{p}_{i,j}^{'} r_j \) is learned in the “…” part of the proof above the clause \( \overline{p}_{1,m}, \overline{p}_{1,m-1}, \ldots, \overline{p}_{i,j} \) on the right hand side of Figure 4. For \( i = 1, 2, n \) and \( j \neq m \), the idea is that the clause \( \overline{p}_{i,j}^{'} r_j \) is learned in the “…” part of the proof above the clause containing \( \overline{p}_{i,j}^{'} \). There are various exceptions to this idea, and several complications, as discussed below. However, in all cases, when working in the subproof in the “…” part...
of the proof above the clause containing $p_{i,j}$, all clauses of the form $p_{i',j'}, r_{j'}$ with $i' > i$ have already been learned. To maintain the regularity property, this subproof must itself be regular, and will not resolve on any literals $p_{i',j'}$ with $i' \leq i$. 

Figure 3: The “skeleton” of the regWRTL proof.
We now outline how a clause \( p_{i,j}, r_j \) is learned in the most typical case, where \( i = 3, \ldots, n-1 \) and \( j \leq m-2 \) or \( i = n \) and \( j \leq m-1 \). (The restriction that \( j < m-2 \) when \( i < n-1 \) is made because a more complicated construction will be needed when \( j = m-1 \) in order to also learn \( p_{i,m}, r_m \).) The part of the proof directly above \( p_{1,m}, \overline{p}_{i-1,m}, \overline{p}_{i,m} \) is presented in Figure 5.

The clause \( \overline{p}_{i,j}, r_j \) on the left hand side of Figure 5 is the clause we want to learn. This clause is derived, and learned, by the derivation given by Theorem 2.2. If \( i' \) and \( i'' \) are the two predecessors of \( i \) in \( G \), then \( i' > i \) and \( i'' > i \) and thus the clauses \( \overline{p}_{i',j'}, r_{j'} \) and \( \overline{p}_{i'',j''}, r_{j''} \) will have all already been learned, so Theorem 2.2 is applicable.

On the right hand side of Figure 5 we do not need to learn anything. We only need to make sure that the right hand side is a well-formed proof. The notation \([p_{i,j},]r_j, \overline{p}_{i-1,m}, r_m\) indicates that \( p_{i,j} \) may not be present. In fact, \( p_{i,j} \) is present exactly when \( i \) is a predecessor of \( i-1 \); otherwise, it is absent.

To describe the right hand side of Figure 5 first suppose that vertex \( i \) is not a predecessor of vertex \( i-1 \) in the graph. In this case, the literal \( \overline{p}_{i,j} \) is not present, and the leaf clause \( \overline{p}_{j}, \overline{p}_{i-1,m}, r_m \) can be proved by the proof given by Theorem 2.3. Second, suppose

\[
\begin{align*}
\overline{p}_{i,m} & \rightarrow \overline{p}_{i,m}.
\end{align*}
\]
We now give a regRTI refutation \( R \) for the \( \text{Stone}(G, m) \) principles. We describe \( R \) by building it from the bottom up, constructing \( R \) in a left-to-right depth-first fashion. At each point of the construction, \( R \) is a partially formed regRTI refutation. The leaves of \( R \) are designated as either “finished” or “unfinished”, and all finished leaves are to the left of all unfinished leaves. The finished leaves are learned clauses; namely, they are either valid Stone clauses or have been derived by an input subderivation earlier in the postorder.
of \( R \). Clauses \( \overline{p}_{i,j}, r_j \), or vertices \( i \) are defined to be \( K \)-learned, \( K = 0, 1, 2, 3 \), according to whether they have been \( K \)-learned in \( R \) at the point of reaching the leftmost unfinished leaf of \( R \).

Each unfinished leaf will contain a clause \( C \) of the form

\[
\overline{p}_{i_1,j_1}, \overline{p}_{i_2,j_2}, \ldots, \overline{p}_{i_k,j_k};
\]

for \( k \geq 0 \). The domain, \( \text{dom}(C) \), of \( C \) is equal to \( \{i_1, \ldots, i_k\} \). Clauses of the form (4.1) will be required to have distinct values for the \( i_i \)'s; for convenience we assume that

\[
i_1 < i_2 < \cdots < i_k.
\]

We let \( \max \text{dom}(C) \) denote the maximum member of \( \text{dom}(C) \), namely \( i_k \). We let \( \text{dom}(C_{\text{pool}}) \) denote the set of \( i \) such that some \( \overline{p}_{i,j} \in C_{\text{pool}} \).

We define the notion of a “well-formed unfinished clause” momentarily. The intuition behind an unfinished leaf \( C \) is based on the idea that a proof is being constructed from the bottom up, by a search process which sets the values of resolution variables in such a way that the clause reached at a given point becomes false. Upon reaching \( C \), the search process has set all of the literals in \( C \) (and \( C_{\text{pool}} \)) false, so that each vertex \( i \in \text{dom}(C) \) has been pebbled with stone \( j_i \). The search process will generate a derivation of \( C \) by proving that each such stone \( j_i \) is red, namely that \( r_j_i \) is true. Since well-formed initial clauses will have \( i_1 = 1 \), this will yield a contradiction and thereby the desired refutation.

The stone \( j_i \) can be shown to be red using one of the following three scenarios: (1) \( i_i \) is a source vertex in \( G \), (2) there is \( j_i' > i_i \) such that \( j_i' = j_i \) and stone \( j_i' \) is red, or (3) the two predecessors of \( i_i \) in \( G \) are pebbled with red stones. Accordingly, for a fixed clause \( C \) of the form (4.1) satisfying (4.2), we define:

**Definition 4.1.** Let \( 1 \leq \ell \leq k \). The vertex \( i_\ell \) is said to be bypassed if there is some \( \ell' > \ell \) such that \( j_{i_{\ell'}} = j_{i_\ell} \). For the maximum such value \( \ell' \), the vertex \( i_{\ell'} \) is called the max-bypasser of \( i_\ell \).

Now let \( 1 \leq \ell < \ell' \leq k \). We say that vertex \( i_{\ell'} \) directly supports vertex \( i_\ell \) if either (1) \( i_{\ell'} \) is the max-bypasser of \( i_\ell \), or (2) \( i_{\ell'} \) is not bypassed and \( i_{\ell'} \) is one of the two predecessors of \( i_\ell \) in \( G \). Note that \( i_\ell \) can directly support multiple \( i_{\ell'} \)'s.

The “supports” relation is the reflexive, transitive closure of “directly supports”; namely, if \( i_{\ell_1} > i_{\ell_2} > \cdots > i_{\ell_s}, \) \( s \geq 1 \), and each \( i_{\ell_q} \) directly supports \( i_{\ell_{q-1}} \), then \( i_{\ell_1} \) supports \( i_{\ell_s} \). We use \( S_\ell(C) \) to denote the set of vertices in \( \text{dom}(C) \) which support \( i_\ell \). Note that \( S_\ell(C) \subseteq \{i_\ell, \ldots, i_k\} \).

The construction of \( R \) starts with the empty clause as the first unfinished clause. The first step will be to generate new unfinished clauses of the form \( \overline{p}_{k,j_1} \); namely of the form (4.1) with \( k = 1 \) and \( i_1 = 1 \). In subsequent steps, an unfinished clause is extended by adding literals \( \overline{p}_{i_{k+1},j_{k+1}} \), where \( i_{k+1} \) is the least vertex \( > i_k \) which supports the vertex \( i_1 = 1 \). The next definitions make this formal.

**Definition 4.2.** A clause \( C \) is a well-formed unfinished clause provided \( C \) is of the form (4.1), satisfies (4.2), with \( i_1 = 1 \) and \( i_k \leq n \), and the following three conditions hold:

i: \( C_{\text{pool}} \) contains only literals of the form \( \overline{p}_{i,j} \) for \( i \leq i_k \), and for each \( i \) there is at most one \( j \) such that \( \overline{p}_{i,j} \) in \( C_{\text{pool}} \).

ii: Let \( 1 \leq \ell \leq k \), and consider \( i_\ell \). Then either

a: The vertex \( i_\ell \) is bypassed, or
b: The vertex $i_\ell$ is not bypassed, and each predecessor $i'_\ell$ of $i_\ell$ in $G$ satisfies one of the following three conditions:

$(\alpha)$: $i'_\ell \in \text{dom}(C)$;  

$(\beta)$: Vertex $i'_\ell$ is already 3-learned, and $i'_\ell \notin \text{dom}(C_{\text{pool}})$; or  

$(\gamma)$: $i'_\ell > i_{k+1}$, and $i'_\ell$ is not 3-learned. 

In case $(\alpha)$, $i'_\ell$ may or may not be 3-learned.

iii: Each $i_\ell \in \text{dom}(C)$ supports $i_1 = 1$. Equivalently, $S_1(C) = \text{dom}(C)$.

The empty clause is also a well-formed unfinished clause.

**Definition 4.3.** A non-empty well-formed unfinished clause of the form (4.1) is extendible provided there is some $i_\ell$ with at least one predecessor $i'_\ell$ in $G$ that satisfies condition $(\gamma)$. The empty clause is also extendible.

During the construction of $R$, all unfinished leaves will be well-formed unfinished clauses. To describe the construction, we explain how to handle the leftmost unfinished leaf of the so-far constructed portion of $R$.

**The extendible case.** First suppose that the leftmost unfinished leaf is an extendible clause $C$ of the form (4.1) with $k > 0$. Considering all vertices $i_\ell$, find one with the least predecessor $i'_\ell$ that satisfies $(\gamma)$. This least $i'_\ell$ is denoted $i_{k+1}$. In the special case where $k = 0$ and $R$ contains just the empty clause (as $C$), let $i_{k+1} = 1$. In either case, $i_{k+1}$ is not 3-learned and not a source vertex for $G$, so $i_{k+1} \leq n$.

With $i_{k+1}$ chosen, define $D_t$ to be the clause $C_\ell = \prod_{i_{k+1}}$. The idea is that we would like to replace $C$ in $R$ with a derivation of $C$ from the Stone clause $p_{i_{k+1},1}, \ldots, p_{i_{k+1},m}$ and the clauses $D_t$ for $t = 1, \ldots, m$ by resolving on the variables $p_{i_{k+1},t}$. The problem with this is that the $D_t$’s may not be well-formed unfinished clauses. So, we consider the set $S_1(D_t)$, namely the set of literals that support the root vertex 1 in $D_t$.

**Claim 4.4.** $i_{k+1} \in S_1(D_t)$.

The claim is trivial for $k = 0$. To prove it when $k > 0$, first suppose that $t$ is equal to some $j_\ell$ for $\ell \leq k$. Choose $i_\ell$ to be the least value such that $t = j_\ell$; of course $i_{k+1}$ is a max-bypasser for $i_\ell$ in $D_t$. Using the fact that $S_1(C)$ contains every $i'_\ell$ for $\ell' \leq k$, a simple induction argument proves that $i'_\ell \in S_1(D_t)$ for every $\ell' \leq \ell$. It follows that $i_{k+1} \in S_1(D_t)$ since it is the max-bypasser of $i_\ell$. Second, suppose that $t$ is distinct from all the $j_\ell$ values. A similar induction argument proves readily that $S_1(D_t)$ contains every $i_\ell$ for $\ell \leq k$. Hence $i_{k+1} \in S_1(D_t)$ since $i_{k+1}$ is a predecessor of some non-bypassed $i_\ell \in S_1(D_t)$.

**Claim 4.5.** There is a $t$ such that $S_1(D_t) = \{i_1, \ldots, i_k, i_{k+1}\}$.

To prove this, take $t$ distinct from all $j_\ell$ values, and use the result from the second subcase of the previous claim. There must exist such a $t$ since $m \geq N$, i.e., there are at least as many stones as vertices.

Define $C_t$ to be the clause containing the literals $\overline{p}_{i_\ell,j_\ell}$ for $i_\ell \in S_1(D_t)$ with $1 \leq \ell \leq k$. Claim 4.5 shows that $C = \bigcup_t C_t$. With the aid of Claim 4.4 and the fact that $C$ satisfies conditions ii and iii of the definition of well-formedness, it follows from the definition of $C_t$ that the clause $C_t, \overline{p}_{i_{k+1},t}$ also satisfies the conditions ii and iii.

Now replace the clause $C$ in $R$ with the resolution derivation shown in Figure 6. The clauses $C_t, \overline{p}_{i_{k+1},t}$ are clearly well-formed unfinished clauses, the $C_t$’s are subclauses of $C$. 


Figure 6: The proof in the extendible case. The resolution variables are the $p_{i_k+1,*}$’s. Each clause $C_i^*$ is the union of the clauses $C_1, \ldots, C_i$.

and the (sub)clauses $C_j^*$ in the figure are defined to equal $\bigcup_{t \leq j} C_t$. By Claim 4.5, $C_m^*$ is equal to $C$, which implies that the inference used to derive $C$ is a valid non-degenerate resolution inference. For all the other newly added inferences, this is clear.

That completes the construction in the case where the leftmost unfinished leaf is extendible.

The non-extendible case. The construction for the case where the leftmost unfinished leaf is not extendible is summarized in the following lemma.

**Lemma 4.6.** Suppose that clause $C$, of the form (4.1), is the leftmost unfinished leaf of $R$, and that $C$ is not extendible. Then there is a regRTI derivation $R_C$ of $C$ from the Stone clauses and the learned clauses in $R$ to the left of $C$. If the clause $p_{i, j}$ was already $K$-learned for $K < 3$, then it becomes $(K+1)$-learned after the clause $C$ is replaced in $R$ with $R_C$. The only variables used as resolution variables in $R_C$ are variables $r_j$, and variables $p_{i, j}$ for $i \notin \text{dom}(C_{\text{pool}})$. The size of $R_C$ is $O(Nm^2)$.

**Proof.** First suppose that $k = 1$. Then $i_1 = 1$, so $p_{i_1, j_1}, \overline{p}_{i_2}$ is a Stone clause. Since $C$ is not extendible, the two predecessors $i'$ and $i''$ of $i_1$ are 3-learned. Theorem 2.2 gives a resolution derivation $L_1$ of $p_{i_1, j_1}, r_{j_1}$ from 3-learned clauses using resolution on (at most) the variables $r_j$ with $j \neq j_1$ and the variables $p_{i', j}$ and $p_{i'', j}$. Form the derivation $R_C$ as

\[
\begin{array}{c}
L_1 \\
\vdots \vdots \\
\overline{p}_{i_1, j_1}, r_{j_1} & \overline{p}_{i_1, j_1}, \overline{r}_{j_1} \\\n\overline{p}_{i_1, j_1} \\
\end{array}
\]

(4.3)

Here and in the sequel, notation of the form “$\vdots$” indicates that the dots should be replaced by the derivation $L$ without its (already pictured) final clause.

\[
\begin{array}{c}
p_{i_{k+1,1}}, \ldots, p_{i_{k+1,m}} \\
C_1, \overline{p}_{i_{k+1,1}} \\
C_1, p_{i_{k+1,2}}, \ldots, p_{i_{k+1,m}} \\
C_2, \overline{p}_{i_{k+1,2}} \\
C^*_2, p_{i_{k+1,2}}, \ldots, p_{i_{k+1,m}} \\
\vdots \\
C^*_{m-2}, p_{i_{k+1,m-1}}, p_{i_{k+1,m}} \\
C_{m-1}, \overline{p}_{i_{k+1,m-1}} \\
C^*_{m-1}, p_{i_{k+1,m}} \\
C_m, p_{i_{k+1,m}} \\
C \\
\end{array}
\]
By the second half of Theorem 2.2, if \( \overline{p}_{i_1,j_1}, r_{j_1} \) was \( K \)-learned for \( K < 3 \), it becomes \( (K+1) \)-learned. The size of \( L_1 \) is \( O(m^2) \).

We now assume that \( k > 1 \). This case is more involved, and takes up most of the rest of the paper.

Define \( B \) to be the set of those non-bypassed vertices \( i_\ell \in \text{dom}(C) \) that have at least one predecessor outside of \( \text{dom}(C) \). Note that since \( C \) is not extendible, a non-bypassed element of \( \text{dom}(C) \) belongs to \( B \) exactly if it has a predecessor satisfying condition \((\beta)\) from the definition of well-formedness; in particular, the predecessor has to be \( 3 \)-learned. We further split \( B \) into the sets \( B_1 \) and \( B_2 \) depending on whether one or both predecessors satisfy \((\beta)\). Note that \( i_1 = 1 \) might be in \( B_1 \), but it cannot be in \( B_2 \), because \( i_2 \) is either a predecessor or a max-bypasser of \( i_1 \). On the other hand, \( i_k \) must be in \( B_2 \). We also define \( B^+ \) to be the set containing \( B \) and all those \( i_\ell \) which are bypassed by a max-bypasser \( i_{\ell'} \in B \). Thus, a vertex \( i_\ell \) is in \( B^+ \) if, according to the minimal partial assignment falsifying clause \( C \), \( i_\ell \) is pebbled by a stone that also covers some vertex in \( B \).

The idea behind the construction of \( R_C \) is that vertices \( i_\ell \) that belong to \( B \cup \{1\} \) determine a “partition” of \( G|_{\text{dom}(C)} \) into possibly overlapping subgraphs (namely the sets \( S_B^C(C) \) defined next). \( R_C \) will deal with these subgraphs independently. The set \( S_B^C(C) \) is a subgraph with sink \( i_k \):

**Definition 4.7.** For \( i_\ell \in \text{dom}(C) \), the set \( S_B^C(C) \) is the smallest set containing \( i_\ell \) and satisfying the following whenever \( i_{\ell'} \in S_B^C(C) \) and either \( \ell' = \ell \) or \( i_{\ell'} \notin B \): (1) If \( i_{\ell'} \) is bypassed in \( C \) by max-bypasser \( i_{\ell''} \notin B \), then \( i_{\ell''} \in S_B^C(C) \) and (2) if \( i_{\ell'} \) is not bypassed in \( C \) and \( i_{\ell''} \) is a predecessor of \( i_{\ell'} \) in \( G \), then \( i_{\ell''} \in S_B^C(C) \).

Note that the closure condition (1) does not allow \( i_{\ell''} \in B \), whereas (2) does allow it.

Enumerate \( B \cup \{1\} \) in decreasing order as \( i_{t_1}, i_{t_2}, \ldots, i_{t_r} \), so that \( t_1 = k \) and \( t_r = 1 \). Below, we use the convention that the index \( q \) ranges over \( 1, \ldots, r \), so that the vertices \( i_{t_q} \) are the members of \( B \cup \{1\} \).

The overall structure of the derivation \( R_C \) is shown in Figure 4 below. \( R_C \) will be built around certain clauses related to the vertices \( i_{t_q} \in B \cup \{1\} \) (the clauses \( C_{q-1}^q, F_q \) in Figure 7). To complete \( R_C \), we will have to construct derivations of the clauses to the side of this “skeleton” (the derivations \( L_{t_q} \) of the clauses \( C_{t_q}^q, F_q, r_{j_{t_q}} \) in Figure 7). In the case where \( i_{t_q} \in B_2 \), this is easy (Lemma 4.12 below), and this easy case is the one which makes \( i_{t_q} = i_k \) become \( (K+1) \)-learned instead of \( K \)-learned. In the case where \( i_{t_q} \in B_1 \cup \{1\} \), we only need to obtain a valid derivation, but this requires a relatively complex construction based on the structure of the subgraph determined by \( i_{t_q} \) (Lemma 4.13).

A clause appearing in \( R_C \) will contain some literals of the form \( \overline{p}_{i,j} \), some literals of the form \( \overline{r}_j \), and at most one literal of the form \( r_j \). These literals have to be selected so as to avoid irregularities and degenerate inferences. \( R_C \) is defined using four special types of clauses: \( C_\ell, C_\ell^*, E_\ell, \) and \( F_\ell \). (The \( C_\ell \)'s are different from the \( C_\ell^* \)'s used for the extendible case.) The \( C_\ell^* \)'s and \( C_\ell^* \)'s consist of \( \overline{p}_{i,j} \)'s, while the \( E_\ell \)'s and \( F_\ell \)'s consist of \( \overline{r}_j \)'s. As suggested by the notation, the \( C_\ell^* \)'s and \( F_\ell \)'s are parametrized by vertices \( i_{t_q} \in B \cup \{1\} \), whereas the \( C_\ell \)'s and \( E_\ell \)'s are parametrized by vertices \( i_\ell \in \text{dom}(C) \).

**Definition 4.8.** If \( i_\ell \) is not bypassed in the clause \( C \), then \( C_\ell \) is the clause containing the literals \( \overline{p}_{i_{\ell'},j_{\ell'}} \) for \( i_{\ell'} \in S_B^C(C) \). If \( i_\ell \) is bypassed in \( C \), then \( C_\ell \) is the same set except the literal \( \overline{p}_{i_{\ell'},j_{\ell'}} \) is omitted.

**Definition 4.9.** \( C_q^* \) equals \( \{ \overline{p}_{i,j} \} \cup \bigcup_{q' > q} C_{t_{q'}} \).
The reason we have to explicitly include $\overline{p}_{1,j_1}$ in $C_q^*$ is that $\overline{p}_{1,j_1} \notin C_1$ if vertex 1 is bypassed.

**Definition 4.10.** If $i_\ell \in B$, then the clause $E_\ell$ is the set of literals $\overline{p}_{j,\ell'}$ such that $\ell' \neq \ell$ and $i_{\ell'} \in S_\ell^B(C) \cap B^+$. For $i_\ell \notin B$, the clause $E_\ell$ contains the literals $\overline{p}_{j,\ell'}$ such that $i_{\ell'} \in S_\ell^B(C) \cap B^+$.

Informally, $E_\ell$ contains the literals $\overline{p}_{j,\ell'}$ for $i_{\ell'} \in B^+$ a source (leaf) vertex above $i_\ell$ relative to the subgraph $S_\ell^B(C)$. The cases where $i_\ell$ is bypassed by a max-bypasser $i_{\ell'}$ deserve special mention. Of course, $j_\ell = j_{\ell'}$. If $i_{\ell'} \in B$, then $i_\ell \in B^+ \setminus B$, and we have $S_\ell^B(C) = \{i_\ell\}$, $C_\ell = \emptyset$, and $E_\ell = \{\overline{p}_{j,\ell'}\}$.

On the other hand, if $i_{\ell'} \notin B$, then $i_\ell \notin B^+$, and we have $S_\ell^B(C) = \{i_\ell\} \cup S_\ell^B(C)$, $C_\ell = C_{\ell'}$, and $E_\ell = E_{\ell'}$.

Before defining the $F_q$’s, we prove a lemma listing some basic properties of the $C$’s, $C^*$’s, and $E$’s:

**Lemma 4.11.**

(a): Each $i_\ell \neq 1$ is a member of some $S_{i_q}^B(C)$. Thus, $C_0^*$ is equal to $C$.

(b): If $1$ is bypassed by max-bypasser $i_\ell$, then $\ell = 2$.

(c): $\overline{p}_{j_{t_{r-1}}} \in E_1$.

(d): $E_{t_q} \subseteq \{\overline{p}_{j_1}, \ldots, \overline{p}_{j_{t_q-1}}\}$, for all $q$.

**Proof.** (a) Given $\ell \neq 1$, consider a chain $i_\ell = i_{\ell_1} > i_{\ell_2} > \cdots > i_{\ell_s} = 1$ of directly supporting vertices from $i_\ell$ to 1. Let $t_\alpha$ be the first member of $B \cup \{1\}$ in this sequence. Then $t_\alpha = t_q$ for some $q$, and we have $i_t \in S_{i_q}^B(C)$. Thus, $\overline{p}_{i_q,j_\ell}$ is in $C_{i_q}^*$ and hence $C_0^*$.

(b) Suppose that 1 has max-bypasser $i_\ell$ with $\ell > 2$. Then $i_\ell > i_2$. There must exist a chain of directly supporting vertices from $i_2$ to 1, but this contradicts $i_\ell > i_2$.

(c) Consider a chain $i_{t_{r-1}} = i_{\ell_1} > i_{\ell_2} > \cdots > i_{\ell_s} = 1$ of directly supporting vertices from $i_{t_{r-1}}$ to 1. By the descending order of the $t_q$’s, none of $i_{t_2}, i_{t_3}, \ldots, i_{t_1}$ is in $B$. If $i_{t_{r-1}}$ is not the max-bypasser of $i_{t_2}$, then $i_{t_{r-1}} \in S_{i_q}^B(C)$ and thus $\overline{p}_{j_{t_{r-1}}} \in E_1$. Suppose instead that $i_{t_{r-1}}$ is the max-bypasser of $i_{t_2}$, so $i_{t_2} \in B^+ \setminus B$. Then $i_{t_2} \in S_{i_q}^B(C)$, so $\overline{p}_{j_{t_2}} \in E_1$. By the definition of bypasser, $j_{t_2} = j_{t_{r-1}}$, so $\overline{p}_{j_{t_{r-1}}} \in E_1$.

(d) Let $\overline{p}_{j_\ell} \in E_{t_q}$, so $\ell \geq t_q$ and $i_\ell \in B^+$. If $i_\ell \in B$, then $\ell > t_q$, and thus $\ell = t_q'$ for some $q' < q$. If $i_\ell \in B^+ \setminus B$, then $i_\ell$ has max-bypasser $i_{\ell'} \in B$. Then $\ell' > \ell \geq t_q$ and thus $\ell'$ equals $t_q$ for some $q' < q$, and by the definition of bypasser $j_{\ell'} = j_\ell$. Therefore $\overline{p}_{j_\ell}$ is the same as $\overline{p}_{j_{t_q'}}$. □

Define $F_q$ to be the clause

$$F_q := \begin{cases} E_{t_q} = E_1 & \text{if } q = r \text{ and } 1 \notin B \\ \overline{p}_{j_1}, \ldots, \overline{p}_{j_{t_{q-1}}} & \text{otherwise} \end{cases}$$

By Lemma 4.11(d), $E_{t_q} \subseteq F_q$. Note that $F_1 = \emptyset$, because $r \neq 1$ and so in evaluating $F_1$ we use the second case of the definition of $F_q$.

We can now describe the derivation $R_C$ in detail. As mentioned, the general structure of $R_C$ is shown in Figure 7. The parts of the derivation displayed in brackets are omitted if $1 \in B^+ \setminus B$.

Note that $C_0^*, F_1$, the final clause of $R_C$, is the same as $C$. For $q < r$, the inference
Again, the final inference deriving \(C_{r-1}^*, F_r\) is non-degenerate.

The upper right clause \(\overline{p}_{1,j_1}, \overline{r}_{j_1}\) is a Stone clause. The derivation \(L_1\) is described in Lemma 4.13.

On the other hand, if \(1 \notin B^+ \setminus B\) and thus \(1\) has max-bypasser \(i_2 \in B\), then \(C_{r-1}^*\) is \(\overline{p}_{1,j_1}\) and \(F_r = E_1\) is \(\overline{r}_{j_1}\). Therefore clause \(C_{r-1}^*, F_r\) is equal to the Stone clause \(\overline{p}_{1,j_1}, \overline{r}_{j_1}\), and there is no need to add to \(R_C\) anything above it.

To finish the description of \(R_C\), we must describe the derivations \(L_{t_q}\). This is done by the next two lemmas.

**Lemma 4.12.** Suppose that \(i_{t_q} \in B_2\). Then there is a regRTI proof \(L_{t_q}\) of \(C_{t_q}, F_q, r_{j_{t_q}}\) of size \(O(m^2)\). The variables used as resolution variables in \(L_{t_q}\) are (at most) the variables \(r_j\) where \(\overline{r}_j \notin F_q \cup \{\overline{r}_{j_{t_q}}\}\) and the variables \(\overline{p}_{i',j}\) and \(\overline{p}_{i'',j}\) for \(i'\) and \(i''\) the predecessors of \(i_{t_q}\) in \(G\).

In the special case of \(q = 1\), so \(t_1 = k\), if \(\overline{p}_{i_k,j_k}, r_{j_k}\) was \(K\)-learned for \(K < 3\), then \(\overline{p}_{i_k,j_k}, r_{j_k}\) becomes \((K+1)\)-learned in \(L_k\).

Note that the values \(i'\) and \(i''\) are not in \(\text{dom}(C_{\text{pool}})\) by the definition of \(B_2\); therefore, the resolution variables \(\overline{p}_{i',j}\) and \(\overline{p}_{i'',j}\) do not violate the regularity condition. Also note that, since \(1 \notin B_2\), we have \(t_q \neq 1\) and the condition that \(\overline{r}_j \notin F_q \cup \{\overline{r}_{j_{t_q}}\}\) is equivalent to \(j \notin \{j_1, \ldots, j_{t_q}\}\). This is precisely what is needed to ensure regularity.

**Proof.** The clause \(C_{t_q}\) consists of the single literal \(\overline{p}_{i_{t_q}, j_{t_q}}\). Thus the derivation \(L_{t_q}\) is the derivation given by Theorem 2.23 When \(q = 1\) and \(t_q = k\), we have \(F_1 = \emptyset\) and \(C_k\) is the clause \(\overline{p}_{i_k,j_k}\). In this case, \(L_k\) is the derivation given by the second part of Theorem 2.23.

**Lemma 4.13.** Suppose that \(1 \leq q \leq r\), and \(i_{t_q} \in B_1 \cup \{1\}\) and \(i_{t_q} \notin B^+ \setminus B\). Let \(N_q = |S_{i_{t_q}}^B(C)|\).

(a): There is a regular dag-like derivation \(L'_{t_q}\) of size \(O(N_q)\) which contains each clause \(C_{i}, E_{\ell}, r_{j_{\ell}}\) for \(i_\ell \notin B^+\) such that \(i_\ell \in S_{i_{t_q}}^B(C)\). The subderivation \(L'_\ell\) of \(L'_{t_q}\) that ends with \(C_{i_\ell}, E_{\ell}, r_{j_{\ell}}\) uses as resolution variables precisely the variables \(r_{j_{\ell'}}\) such that \(j_\ell \neq j_{\ell'}\), \(i_{\ell'} \in S_{i_{t_q}}^B(C)\), and \(i_{\ell'} \notin B^+\).

If \(i_{t_q} \notin B_1\), then \(t_q = 1\) and \(q = r\) and the final clause of \(L'_{t_q}\) is \(C_1, E_1, r_{j_1}\) (that is, with \(\ell = t_q = 1\)). Suppose instead that \(i_{t_q} \in B_1\) and \(i_\ell\) is the (only) predecessor of \(i_{t_q}\).
such that $i_\ell \in \text{dom}(C^\text{pool})$. If $i_\ell \in B^+$, then $S_{i_q}^B(C) \subseteq B^+$ and $L'_{t_q}$ is empty. If $i_\ell \notin B^+$, then the final clause of $L'_{t_q}$ is $C_\ell, E_\ell, r_{j_\ell}$.

(b): There is a regRTI derivation $L_{t_q}$ of $C_{t_q}, F_q, r_{j_{t_q}}$ of size $O(N_q^2 + m)$. The variables used as resolution variables in $L_{t_q}$ are at most the variables $r_j$ where $\overline{r}_j \notin F_q \cup \{\overline{r}_{i_q}\}$ and, if $i_{t_q} \in B_1$, the variables $p_{\ell', j}$ where $i''$ is the predecessor of $i_{t_q}$ such that $i'' \notin \text{dom}(C^\text{pool})$.

Note that $i_{t_q} \in B^+ \setminus B$ only if $t_q = 1$, and this is the case where $L_1$ is not needed.

**Proof.** The regular dag-like refutation $L_{t_q}$ for part (a) is constructed by induction on $\ell$ such that $i_\ell \in S_{i_q}^B(C) \setminus B^+$. We add the clauses $C_\ell, E_\ell, r_{j_\ell}$ to $L'_{t_q}$ for larger values of $\ell$ first, making sure that the condition on resolution variables remains satisfied. The inductive argument splits into four cases.

**Case 1:** $i_\ell$ is bypassed by max-bypassers $i_{\ell'}$. Since $i_\ell \notin B^+$, we have $i_{\ell'} \notin B^+$. The induction hypothesis tells us that $C_{\ell'}, E_{\ell'}, r_{j_{\ell'}}$ appears in the already constructed portion of $L'_{t_{\ell'}}$. From the remarks after the definitions of $C_\ell$ and $E_\ell$, we have $j_\ell = j_{\ell'}$ and $C_\ell = C_{\ell'}$ and $E_\ell = E_{\ell'}$. Thus $C_{\ell'}, E_{\ell'}, r_{j_{\ell'}}$ is exactly the same clause as $C_\ell, E_\ell, r_{j_\ell}$. So no further resolution inferences need to be added to $L'_{t_q}$ to handle $i_\ell$, and $L'_{t_q} = L'_{t_{\ell'}}$. Since $S_{i_q}^B(C) = \{i_{\ell'}\} \cup S_{i_{\ell'}}^B(C)$ and $i_\ell \notin B^+$, the subderivation $L'_{t_q}$ satisfies the condition about which resolution variables are used.

The remaining cases all assume that $i_\ell$ is not bypassed.

**Case 2 (the base case):** both of $i_\ell$’s predecessors $i_{\ell'}$ and $i_{\ell''}$ are in $B^+$. Then $C_\ell$ is $\overline{p}_{i_{\ell'}}, \overline{p}_{i_{\ell''}}, p_{i_{\ell'}}, \overline{p}_{i_{\ell''}}$. Also, $E_\ell$ is $\overline{p}_{j_{\ell'}}, \overline{p}_{j_{\ell''}}$. Therefore the Stone clause [14,4] is the same as $C_\ell, E_\ell, r_{j_\ell}$, so $L'_{t_q}$ consists of just this clause.
Case 3: neither of $i'_{\ell}$'s predecessors $i'_{\ell'}$ and $i''_{\ell'}$ is in $B^+$. The already constructed part of $L'_{\ell}$ contains the clauses $C_{\ell'}, E_{\ell'}, r_{j_{\ell'}}$, and $C_{\ell''}, E_{\ell''}, r_{j_{\ell''}}$. Assume for the moment that $j_{\ell'} \neq j_{\ell''}$. W.l.o.g., the highest index $s''$ such that $j_{s''} = j_{\ell'}$ is strictly greater than highest index $s'$ such that $j_{s'} = j_{\ell''}$; otherwise interchange $\ell'$ and $\ell''$. It follows from the induction step for Case 1 that $L'_{\ell''}$ equals $L'_{s''}$. Therefore, by the assumption that $s' < s''$ and the inductive condition on resolution variables, $r_{j_{\ell'}}$ is not resolved on in $L'_{\ell''}$.

Using the Stone clause

$$\overline{p}_{i'_{\ell} j_{\ell'}, \overline{p}_{i'_{\ell'} j'_{\ell'}, \overline{p}_{i'_{\ell''} j''_{\ell'}}} \overline{p}_{i'_{\ell''}}, r_{j_{\ell'}} \overline{L'_{\ell'}}$$

form $L'_{\ell}$ as

$$\frac{\overline{p}_{i'_{\ell} j_{\ell'}, \overline{p}_{i'_{\ell'} j'_{\ell'}, \overline{p}_{i'_{\ell''} j''_{\ell'}}} \overline{p}_{i'_{\ell''}}, C_{\ell'}, E_{\ell'}, r_{j_{\ell'}} \overline{L'_{\ell'}}}{C_{\ell'}, E_{\ell'}, r_{j_{\ell'}} \overline{L'_{\ell'}}}$$

We have $E_{\ell} = E_{\ell'} \cup E_{\ell''}$. The literals $\overline{p}_{i'_{\ell} j_{\ell'}}$ and $\overline{p}_{i'_{\ell'} j'_{\ell'}}$ may or may not appear in $C_{\ell'}$ and $C_{\ell''}$ (respectively), but in any case $C_{\ell}$ is the same as $\overline{P}_{i'_{\ell} j_{\ell'}, C_{\ell'}, \overline{p}_{i'_{\ell'} j'_{\ell'}}, C_{\ell''}, \overline{p}_{i'_{\ell''}}, C_{\ell'}, E_{\ell'}, r_{j_{\ell'}} \overline{L'_{\ell'}}}$.

Note that the proof as pictured above might be slightly misleading: we are constructing a dag-like derivation, not a tree-like derivation and the subderivations $L'_{\ell}$ and $L'_{\ell''}$ need not be disjoint. The regularity of $L'_{\ell}$ follows from the induction hypotheses and the fact that $r_{j_{\ell'}}$ is not a resolution variable of $L'_{\ell''}$. The condition on which resolution variables are used in $L'_{\ell}$ follows from $S^B_{\ell}(C) = \{i_{\ell'} \} \cup S^B_{\ell'}(C) \cup S^B_{\ell''}(C)$.

For the remaining part of case 3, suppose that $j_{\ell'} = j_{\ell''}$. Let $s$ be the highest index such that $j_{s'} = j_{\ell'} = j_{\ell''}$. By the remarks after the definitions of $C_{\ell}$ and $E_{\ell}$, we have $C_{\ell'} = C_{\ell''} = C_s$ and $E_{\ell'} = E_{\ell''} = E_s$. Thus also $L'_{\ell'} = L'_{\ell''} = L'_s$. Now, argue as in the previous case, but replace the inferences (4.5) with the inference

$$\frac{\overline{p}_{i'_{\ell} j_{\ell'}, \overline{p}_{i'_{\ell'} j'_{\ell'}, \overline{p}_{i'_{\ell''} j''_{\ell'}}} \overline{p}_{i'_{\ell''}}, C_{\ell'}, E_{\ell'}, r_{j_{\ell'}} \overline{L'_{\ell'}}}{C_{\ell'}, E_{\ell'}, r_{j_{\ell'}} \overline{L'_{\ell'}}}$$

The left hypothesis is a Stone clause. The rest of the argument for this subcase is as in the previous case.

Case 4: $i_{\ell}$ has predecessors $i'_{\ell'} \notin B^+$ and $i''_{\ell''} \in B^+$. The induction hypothesis says that the clause $C_{\ell'}, E_{\ell'}, r_{j_{\ell'}}$ has already been derived. Then $C_{\ell} = \overline{P}_{i'_{\ell} j_{\ell'}, \overline{p}_{i'_{\ell'} j'_{\ell'}, \overline{p}_{i'_{\ell''} j''_{\ell'}}} \overline{p}_{i'_{\ell''}}, C_{\ell'}, E_{\ell'}, r_{j_{\ell'}} \overline{L'_{\ell'}}}$, and $E_{\ell}$ is $\overline{p}_{i'_{\ell''}}, E_{\ell'}$, so we can form $L'_{\ell}$ as

$$\frac{\overline{p}_{i'_{\ell} j_{\ell'}, \overline{p}_{i'_{\ell'} j'_{\ell'}, \overline{p}_{i'_{\ell''} j''_{\ell'}}} \overline{p}_{i'_{\ell''}}, C_{\ell'}, E_{\ell'}, r_{j_{\ell'}} \overline{L'_{\ell'}}}{C_{\ell'}, E_{\ell'}, r_{j_{\ell'}} \overline{L'_{\ell'}}}$$

using resolution on $r_{\ell'}$. The regularity of $L'_{\ell}$ and the conditions on which resolution variables are used in $L'_{\ell}$ follow from the induction hypothesis for $L'_{\ell'}$ and the fact that $S^B_{\ell}(C) = \{i_{\ell}, i''_{\ell''} \} \cup S^B_{\ell'}$. 
That completes the proof of part (a). The size bound $O(N_q^2)$ on $L'_q$ follows from the fact that each of the four cases in the construction of $L'_q$ added $O(1)$ clauses.

To apply part (a) in the proof of (b), we need regRTI derivations $L''_{t_q}$, instead of the regular dag-like refutations $L'_{t_q}$. For this, Theorem 3.3 of [BHJ08] states that the desired derivation $L''_{t_q}$, containing exactly the same clauses as $L'_{t_q}$, can be constructed from $L'_{t_q}$ with the size of $L''_{t_q}$ bounded by the product of the size and the height of $L'_{t_q}$. Thus, the size of $L''_{t_q}$ is $O(N_q^2)$. Furthermore, $L''_{t_q}$ uses the same resolution variables as $L'_{t_q}$.

We now prove part (b). First suppose that $q = r$ and $1 \notin B$. Then $t_q = 1$ and $F_q = E_1$. In this case, the regRTI derivation $L''_{t_q}$ obtained from part (a) is already the desired derivation. This only uses resolution variables $r_{j\ell}$ such that $i_{\ell} \notin B^+$ and hence $j_\ell \notin \{j_1, \ldots , j_q\}$.

Otherwise, $i_{t_q} \in B_1$, and hence $F_q = \{\tau_{j_1}, \ldots , \tau_{j_{t_q}}\}$. Let $i_{\ell'}$ and $i''$ be the predecessors of $i_{t_q}$, so, $i''$ is 3-learned and $i'' \notin \text{dom}(C^{\text{pool}})$. Note that $j_{\ell'}$ may or may not be in $\{j_1, \ldots , j_q\}$. Consider the $m - 2$ many Stone clauses for $j'' \notin \{j_1, j_{\ell'}\}$:

$$\overline{F}_{i_{\ell'}}, \tau_{j_{\ell'}}, \tau_{j''}, F_{i_{t_q}}, r_{j_{t_q}};$$

Resolving these with the 3-learned clauses $\tau_{j''}, r_j''$ for $j'' \notin \{j_1, \ldots , j_q, j_{\ell'}\}$ and the Stone clause $\vee_j p_j''$, $r_j''$ gives the clause

$$\overline{F}_{i_{\ell'}}, \tau_{j_{\ell'}}, F_{i_{t_q}}, r_{j_{t_q}};$$

by resolution on the variables $r_{j''}$ for $j'' \notin \{j_1, \ldots , j_q, j_{\ell'}\}$ and the variables $p_j''$, $r_j''$ for all $j''$.

Suppose that $i_{\ell'} \in B^+$ and thus $\tau_{j_{\ell'}} \notin F_q$ and $C_{t_q}$ is the clause $\overline{F}_{i_{\ell'}}, \tau_{j_{\ell'}}, \tau_{j''}, F_{i_{t_q}}, r_{j_{t_q}}$. Then (4.6) is the same as $C_{t_q}, F_q, r_{j_{t_q}}$ and the construction of $L_{t_q}$ for part (b) is complete. In this case, $L_{t_q}$ has size $O(m)$.

Alternately, suppose that $i_{\ell'} \notin B^+$, so $\tau_{j_{\ell'}} \notin F_q$. By the last assertion of part (a), $L''_{t_q}$ is a regRTI derivation of $C_{\ell'}, E_{\ell'}, r_{j_{\ell'}}$ of size $O(N_q^2)$. Form $L_{t_q}$ by resolving this against (4.6) on the variable $r_{j_{\ell'}}$ to obtain $C_{t_q}, F_q, r_{j_{t_q}}$. This is a valid resolution inference since $C_{t_q}$ is $\overline{F}_{i_{t_q}}, \tau_{j_{t_q}}, F_{i_{\ell'}}, \tau_{j_{\ell'}}, C_{\ell'}$, and $E_{\ell'} \subseteq F_q$. The size of $L_{t_q}$ is $O(N_q^2 + m)$.

This completes the description of $L_{t_q}$, and the proof of Lemma 4.13.

This also completes the construction of the derivation $R_C$ of Lemma 4.16. Since the construction of $R_C$ invokes Lemmas 4.12 and 4.13 at most $N$ times, the size of $R_C$ is bounded by $O(Nm^2 + N(N^2 + m)) = O(Nm^2)$.

Finishing the proof. All that remains to finish the proof of Theorem 1.15 is to bound the size of the refutation $R$. As described above, $R$ is built up from a derivation containing only the empty clause by applying the constructions of Figure 6 and Lemma 4.6 always to the leftmost currently unfinished leaf.

If the clause at that leaf is non-extendible, it is dealt with using Lemma 4.6 and no new unfinished leaves appear.

Otherwise, the leftmost unfinished leaf contains an extendible clause, and it is dealt with using the construction of Figure 6 leading to $m$ new unfinished leaves, at least one of them labeled with a clause $C_t, \overline{F}_{i_{t_q}}, \tau_{j_{t_q}}$ such that $\overline{F}_{i_{t_q}}, \tau_{j_{t_q}}$ has not been 3-learned.

A clause of the type allowed to appear in an unfinished leaf can be iteratively extended at most $n$ times, so at some point we have to reach a situation in which the construction of Figure 6 produces only non-extendible clauses. When Lemma 4.6 is applied to those clauses,
at least one \(K\)-learned clause of the form \(\overline{p}_{i,j}, r_j\) becomes \((K+1)\)-learned. There are \(nm\) clauses of the form \(\overline{p}_{i,j}, r_j\) to be learned. Once all of them have been 3-learned, all remaining unfinished leaves become non-extendible and can be dealt with using the construction of Lemma 4.6.

Consider the set of clauses in \(R\) which at some point were unfinished leaves during the construction of \(R\). Call such a clause green if it was handled with Lemma 4.6 and thereby a clause of the form \(\overline{p}_{i,j}, r_j\) became \((K+1)\)-learned instead of \(K\)-learned. The remaining clauses are called non-green: these clauses were either handled by Lemma 4.6 but without any clause \(\overline{p}_{i,j}, r_j\) becoming \((K+1)\)-learned, or were handled with the construction of Figure 6. These green and non-green clauses inherit a tree structure from \(R\). It follows from the discussion above that this tree is \(m\)-branching, has depth at most \(n\), and contains at most \(3nm\) green leaves. Moreover, each node in the tree is a green leaf, is an ancestor or sibling of a green leaf, or is the sibling of an ancestor of a green leaf. It is straightforward to prove that such a tree has at most \(O(n^2m^2)\) leaves. Since each of these leaves corresponds to a single application of Lemma 4.6, the size of \(R\) is at most \(O(N^3m^4)\). This completes the proof of Theorem 1.15.

Since the Stone tautologies contain \(O(Nm^3)\) many symbols, and since \(N \leq m\), the size upper bound \(O(N^3m^4)\) is quadratic in the size of the clauses being refuted.

The refutation \(R\) described above may not be greedy. Although we lack a proof, it is possible that \(R\) can be made greedy by omitting subderivations of already learned clauses. It is also possible that \(R\) is, or could be made to be, unit-propagating. In particular, note that the only unit clauses that appear in \(R\) are the literals \(\overline{p}_{i,j}\) that appear as unfinished clauses in the very first step of the construction of \(R\). However, we have not tried to formally analyze the greedy or unit-propagating properties of \(R\).

ACKNOWLEDGEMENT

We thank the two anonymous referees for useful comments and suggestions.

The proof is complete, If only I’ve stated it thrice.

Fit the Fifth – The Beaver’s Lesson, The Hunting of the Snark

Lewis Carroll

REFERENCES

[AJPU07] Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. Theory of Computing, 3(5):81–102, 2007.

[BB12] Maria Luisa Bonet and Samuel R. Buss. 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, pages 45–57, 2012.

[BBJ14] Maria Luisa Bonet, Samuel R. Buss, and Jan Johannsen. Improved separations of regular resolution from pool resolution and clause learning. Journal of Artificial Intelligence Research, 49:669–703, 2014.

[BHJ08] Samuel R. Buss, Jan Hoffmann, and Jan Johannsen. 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, 2008.
[BKS04] Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, 2004.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, 1962.

[DP60] Martin Davis and Hillary Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7(3):201–215, 1960.

[HBPVG08] Philipp Hertel, Fahim Bacchus, Toniann Pitassi, and Allen Van Gelder. Clause learning can effectively p-simulate general propositional resolution. In *Proc. 23rd AAAI Conf. on Artificial Intelligence (AAAI 2008)*, pages 283–290. AAAI Press, 2008.

[MSS99] João P. Marques-Silva and Karem A. Sakallah. GRASP — A new search algorithm for satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.

[Urq11] Alasdair Urquhart. A near-optimal separation of regular and general resolution. *SIAM Journal on Computing*, 40(1):107–121, 2011.

[VG05] Allen Van Gelder. 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, pages 580–594. Springer-Verlag, 2005.