Algorithms & Complexity Results for Input & Unit Resolution

Alexander Hertel & Alasdair Urquhart*
Department of Computer Science
University of Toronto

February 4, 2008

Abstract

In this paper we explore the complexity of various problems pertaining to Input Resolution, a very restricted form of Resolution which requires that at least one input to every instance of the resolution rule be an input clause. This proof system is of practical interest because Input Resolution is equivalent to Unit Resolution, a popular subroutine used in many areas of computer science, especially SAT-solvers.

We first prove a number of tractability results for Input Resolution, which leads us to algorithms showing its automatizability, as well as the $\mathcal{NP}$-Completeness of the Input Resolution size problem. However, our main result is an exact equivalence between the pure black pebbling number of any DAG G and the Input Resolution total space of the pebbling contradiction formula $Peb^1(G)^*$. This equivalence allows us to prove the $\mathcal{PSPACE}$-Completeness of various forms of the Input Resolution derivation total space problem, and as a corollary yield the $\mathcal{PSPACE}$-Completeness of the Input derivation width problem as well as a massive size / total space tradeoff for Input Resolution. These $\mathcal{PSPACE}$-Completeness and tradeoff results are somewhat surprising when juxtaposed with our earlier theorems; on one hand, Input Resolution seems to be a completely tractable and even trivial Resolution refinement, but on the other hand some of its other characteristics are highly complicated.

1 Introduction & Motivation

The Resolution (RES) proof system has been studied extensively, and is understood quite well. Because it forms the basis of a vast number of automated theorem proving algorithms, there is a strong practical motivation for understanding its limitations. When devising SAT-solving algorithms, there is always a tension between proof size and proof search; more powerful proof systems have shorter proofs, but they are much harder to find, making it difficult to design proof search algorithms. For this reason, virtually all automated theorem provers implement weak proof systems such as RES rather than more powerful systems such as Frege. However, automated proof search is even too complicated in RES, which is why researchers have further developed 'Resolution refinements', restricted forms of Resolution aimed at simplifying proof search at the possible expense of increasing minimum proof size. Common refinements include Tree Resolution / DPLL and clause learning, which are both very successful SAT-solving algorithms. The ultimate goal of automated theorem proving is to develop algorithms capable of finding proofs which are only polynomially larger than the optimal. This property is referred to as 'automatizability'.

One particularly extreme refinement is Input Resolution (I-RES), in which we require that at least one of the inputs to each application of the resolution rule be an initial, or input clause. This clearly restricts the search space that any I-RES algorithm would have to deal with, but the cost is very high, since I-RES is not even complete. However, despite its simplicity and obvious limitations, we shall show that the

*The authors gratefully acknowledge NSERC and the University of Toronto Department of Computer Science for supporting this research.

I-RES proof system is much more interesting than it may first appear. It is important both theoretically as well as practically because of a Theorem by Chang [Cha70] which states that a formula has an I-RES refutation if and only if it has a Unit Resolution (U-RES) refutation. Since U-RES is such an important and widely-used subroutine in SAT-solving as well as other areas, this gives a strong motivation for better understanding I-RES.

In this paper we prove a number of complexity results for I-RES including $\mathcal{P}$-Completness, $\mathcal{NP}$-Completeness, $\mathcal{PSPACE}$-Completeness, and exponential tradoff results. In addition, we show that I-RES is automatizable, and optimally automatizable on minimally unsatisfiable formulas. The first part of this paper contains straightforward results dedicated to investigating the more tractable aspects of I-RES, whereas the second half investigates its more complex characteristics.

Specialized definitions and concepts which we shall require can be found in Section 2. Beyond these definitions, we assume that the reader is familiar with basic proof complexity as well as general complexity theory, and use [CK01, Pap94] as our standard references.

Our results start in Section 3, where we show how formulas with I-RES refutations ($IRES$-$UNSAT$) are related to Horn formulas, and minimally unsatisfiable formulas which have I-RES refutations (MU-$IRES$-$UNSAT$) are related to the minimally unsatisfiable formulas in $MU(1)$.

Next, in Section 4, we prove a number of tractability results for I-RES. For example, we combine some previous results to show that $IRES$-$UNSAT$ is $\mathcal{P}$-Complete. We also show that the problem of determining if a formula is minimally unsatisfiable and has an I-RES refutation, as well as the problem of determining if formula F is minimally unsatisfiable and has an I-RES refutation of size at most k are both in $\mathcal{P}$.

These results lead us to Section 5, in which we develop algorithms for automatizing I-RES and automatizing I-RES optimally on minimally unsatisfiable formulas.

This brings us to the second half of our paper and the more complicated aspects of I-RES. In Section 6, we note that the problem of approximating optimal I-RES and U-RES refutation size to within a linear factor for $IRES$-$UNSAT$ is $\mathcal{NP}$-Hard and that given a formula F and integer k, the problem of determining whether F has an I-RES refutation of size at most k is $\mathcal{NP}$-Complete. This stands in contrast with the tractability of the same problem for MU-$IRES$-$UNSAT$ from the previous section.

The rest of this paper contains our main results, and is dedicated to the study of I-RES space. These results are more complicated our earlier ones. We therefore begin this part of the paper in Section 7 by reviewing some of the most important previous work related to space for resolution-based proof systems.

Next, in Section 8 we prove an equivalence between I-RES total space and pebbling, followed immediately by its implications for complexity theory. More specifically, we show that for any binary DAG G, its black pebbling number (defined with sliding) is off by exactly a constant from the total space required by any I-RES proof of a slight modification to the formula $Peb_1(G)$. This equivalence allows us to prove our main results, namely the $\mathcal{PSPACE}$-Completeness of various forms of the I-RES total space problem, which is proved by reducing from the black pebbling game on DAGs, proven $\mathcal{PSPACE}$-Complete by Gilbert, Lengauer, and Tarjan [GLT80]. These results show that although with respect to some complexity measures, I-RES is very simple, with respect to others it is extremely complex.

Finally, in Section 9 we prove two interesting corollaries to our main results. The first is the $\mathcal{PSPACE}$-Completeness of I-RES derivation width. However, the most interesting corollary that we prove is an extreme (and optimal) size / total space tradeoff for I-RES; we show that there exists an infinite family of formulas whose I-RES proofs with the minimum required total space have size $2^{\Omega(n)}$, where n is the number of distinct variables. However, if only one single additional unit of total space is permitted, then the size drops to only $O(n)$, where n is the number of variables, once again contrasting strongly with the apparent simplicity of I-RES as suggested by the earlier tractability and automatizability results.

2 Definitions

2.1 Resolution Proof, Size, & Width

The notions of size and width can be formalized using a fairly simple definition of what constitutes a RES proof, whereas the notion of space requires a slightly more complicated definition:

Definition 2.1 (RES Proof). *A clause C is a set of literals. If $C \vee x$ and $D \vee \neg x$ are clauses, then the resolution rule allows us to derive the clause $C \vee D$ by resolving on the variable x. If F is a set of clauses, then the sequence of clauses $\pi = C_1, C_2, ..., C_k$ is a RES proof of C_k from F if each C_i in π appears in F (i.e. is an input, or initial clause) or follows from two previous clauses in π by the resolution rule.*

If the graph underlying the structure of π is a tree (i.e. each clause in π is a premise for at most one application of the resolution rule), then the proof is said to be a Tree-Like Resolution (T-RES) proof. Otherwise it is said to be DAG-Like. A RES refutation of F is a RES proof from F in which $C_k = \emptyset$ (the empty clause).

This allows us to define size and width:

Definition 2.2 (Resolution Size & Width). *If a RES proof π of formula F contains k clauses, then it is said to have size k. The width of a clause C refers to how many literals it contains, and is denoted $w(C)$. The width of a formula F is the width of the widest clause in F, and is denoted $w(F)$. The width of a RES refutation π, denoted $w(\pi)$ is equal to the width of its widest clause, and is denoted $w(\pi)$. Finally, the minimum width of any RES refutation of F is denoted $w(F \vdash_{\mathsf{RES}} \emptyset)$.*

2.2 Resolution Space

The definition of the space measure that we shall be discussing requires an alternative definition of RES proof which depends on the notion of configuration:

Definition 2.3 (Configuration-Style RES Proof). *A configuration $\mathbb{C}$ is a set of clauses. If F is a set of clauses, then the sequence of configurations $\pi = \mathbb{C}_0, \mathbb{C}_1, ..., \mathbb{C}_k$ is a RES proof of C from F if $\mathbb{C}_0 = \emptyset$, $C \in \mathbb{C}_k$, and for each $i < k$, $\mathbb{C}_{i+1}$ is obtained from $\mathbb{C}_i$ by one of the following rules:*

1. *deleting one or more of its clauses,*
2. *adding the resolvent of two clauses of $\mathbb{C}_i$,*
3. *adding one or more of the clauses of F (initial clauses).*

In addition, π is said to be an I-RES proof if at least one input to every instance of the resolution rule is an input clause, and it is said to be an I-RES-W$^-$ (I-RES with negative weakening) proof if we add the following rule:

4. replacing one of the clauses $C \in \mathbb{C}_i$ with the clause $C \cup \{\neg x\}$, which was obtained by weakening C with an arbitrary negative literal $\neg x$.

In any type of Resolution proof, the first two clauses resolved on are called the 'top clauses' of π. The final result of the refutation, C_k is called the 'goal clause'. If $\emptyset \in C_k$, then π is a refutation. Finally, if at least one input to every instance of the resolution rule is a unit clause (i.e. it contains just one literal), then we say that π is a Unit Resolution proof.

This leads us to our definition of space. Intuitively, space is the amount of memory required in order to compute π. Note that our terminology differs from that introduced by Ben-Sasson in [BS02]. What Ben-Sasson refers to as 'variable space', we refer to as 'total space':

Definition 2.4 (Total Space). *Let F be a set of clauses and π be a configuration-style* RES *proof of clause C from F. The total space of a configuration $\mathbb{C}$ in π, denoted $TS(\mathbb{C})$ is defined as $\sum_{C \in \mathbb{C}} w(C)$. The total space of π, denoted $TS(\pi)$ is the maximum $TS(\mathbb{C})$ over all $\mathbb{C}$ in π. Finally, the total space of resolving C from F, denoted $TS(F \vdash_{\mathsf{RES}} C)$, is the minimum $TS(\pi)$ over all* RES *proofs π of C from F.*

2.3 Pebbling Circuits & Games

The investigation of RES space is closely associated with the well-known pebbling game and pebbling number of a DAG, originally explored in [Coo73, CS76] as a means of investigating bounds on storage requirements. There are several different versions of pebbling games, but the most popular ones are captured by the following generalized description:

Definition 2.5 (Black-White Pebbling Game on DAGs). *The black-white pebbling game is a single-player game where the goal is to 'pebble' a DAG G in which each node is an AND gate or a source. The leaves of G have in-degree 0 and are referred to as 'source' nodes. A single vertex in G is referred to as a 'target' or 'output' node, and has out-degree 0. All non-target nodes can have arbitrary out-degree, except of course when G is a tree, in which case the maximum out-degree is 1. In all cases, non-source nodes can have arbitrary in-degree. A DAG in which all non-source nodes have in-degree 2 is called a 'binary DAG'. The game involves two different types of pebbles (black and white), and has the following rules:*

1. *The game starts with no pebbles on the DAG.*
2. *A gate may have at most one pebble on it at any time.*
3. *At any point, the player may place any pebble onto a source node or remove any pebble from a source.*
4. *At any point, the player may remove any black pebble from any gate or place a white pebble on any empty gate.*
5. *For any unpebbled node v, if all of a v's immediate predecessors have pebbles on them, then the player may place a black pebble on v. Alternatively, if the sliding rule is present, the player may slide a black pebble from u to v, where u is a predecessor of v.*
6. *For any node v with a white pebble on it, if all of a v's immediate predecessors have pebbles on them, then the player may remove the white pebble from v. Alternatively, if all but one of v's predecessors are pebbled, then the player my slide the white pebble from v to the remaining unpebbled predecessor of v.*
7. *The game ends once the circuit has a black pebble on the target node and there are no white pebbles present on the DAG.*

The rules above define the standard 'black-white pebbling game' on DAGs. We can further restrict the game by disallowing the use of white pebbles, resulting in the standard 'black pebbling game'. The black pebbling game and the black-white pebbling game played on DAGs are the most common forms of the game found in the literature, although pebbling on monotone circuits has also been investigated. This leads us to the definitions of the 'black-white pebbling number' and the 'black pebbling number':

Definition 2.6 (Pebbling Numbers of DAGs). *Given a DAG G, the 'black-white pebbling number' of G, denoted $BW\text{-}Peb(G)$ is the minimum number of total pebbles that the player needs in order to complete the black-white pebbling game on G without violating any of its rules.*

The 'black pebbling number' of G, denoted $B\text{-}Peb(G)$ is the minimum number of black pebbles that the player must be given in order to complete the black pebbling game on G without violating any of its rules.

Note that each version of the pebbling game can either be defined with or without sliding; this is an important point because allowing sliding has a slight effect on the pebbling number (see Lemma 7.1). Another issue is whether sliding is compulsory or optional, i.e. if one is allowed to place a pebble even when sliding is possible. For the purposes of this paper, we will allow sliding unless otherwise stated.

2.4 Pebbling Contradictions

A number of important families of unsatisfiable formulas called the 'pebbling contradictions' are based on the various different forms of the pebbling game. A brief history of how these formulas have been used in the literature is given in [BS02].

Definition 2.7 (One-Colour Pebbling Contradictions for DAGs). *The one-colour pebbling contradiction of a DAG G, denoted $Peb^1(G)$, is an unsatisfiable formula constructed by taking G, and creating the following clauses in which each variable x is interpreted as meaning that vertex x has a pebble on it:*

1. *For each source node s, create the singleton clause $\{s\}$.*
2. *For each node y_0 of degree d with immediate predecessors $y_1, y_2, ..., y_d$, create the propagation clause $\{\neg y_1, \neg y_2, ..., \neg y_d, y_0\}$.*
3. *Finally, for the target node t, create the singleton clause $\{\neg t\}$.*

It should be easy to see that for any DAG G, $Peb^1(G)$ is unsatisfiable; this is because the source node variables must all be true, and all of the propagation clauses send this truth towards the target. The target variable must therefore be true, but the negative singleton clause(s) for the target forces it to be false, ultimately creating a contradiction.

3 Input Resolution, Horn Formulas, and MU Formulas

In this section we define $IRES\text{-}UNSAT$, the set of formulas which have I-RES refutations, as well as $MU\text{-}IRES\text{-}UNSAT$, its minimally unsatisfiable subset, and relate these languages to unsatisfiable Horn formulas and minimally unsatisfiable (MU) formulas. More specifically, we show that $HORN\text{-}UNSAT$ is a proper subset of $IRES\text{-}UNSAT$, and $MU\text{-}IRES\text{-}UNSAT$ is a proper subset of $MU(1)$. Before proceeding, we formally define these languages:

Our first languages are the formulas for which I-RES and U-RES are complete:

Definition 3.1 (IRES-UNSAT & URES-UNSAT). *$IRES\text{-}UNSAT$ is the set of all unsatisfiable formulas which have* I-RES *refutations, and $URES\text{-}UNSAT$ is the set of all unsatisfiable formulas which have* U-RES *refutations.*

Our next set of languages are related to Horn formulas and are defined as follows:

Definition 3.2 (Horn Formulas, HORN-SAT, & HORN-UNSAT). *A formula F is said to be Horn if each of its clauses contains at most one positive literal. We will refer to the set of all satisfiable Horn formulas as $HORN\text{-}SAT$, and the set of all unsatisfiable Horn formulas as $HORN\text{-}UNSAT$.*

It is worth noting that $Peb_1(G) \in HORN\text{-}UNSAT$.

Our final set of languages are based on minimally unsatisfiable formulas:

Definition 3.3 (MU(k), MU-IRES-UNSAT, & MU-HORN-UNSAT). *A formula F is said to be minimally unsatisfiable if it is unsatisfiable, but the same cannot be said of any proper subset of its clauses. These minimally unsatisfiable formulas comprise the set MU. We define $MU(k)$ as the set of all minimally unsatisfiable formulas on n variables which contain exactly $n+k$ clauses. In addition, MU-$IRES$-$UNSAT$ contains all of the minimally unsatisfiable formulas in $IRES$-$UNSAT$, and MU-$HORN$-$UNSAT$ contains all of the minimally unsatisfiable formulas in $HORN$-$UNSAT$.*

Again, it is worth noting that $Peb_1(G) \in MU\text{-}HORN\text{-}UNSAT$ for any DAG G in which every node is essential (i.e. every pebbling of G uses every node in G at some point during the pebbling).

3.1 Relating Input Resolution & Unit Resolution

In this section we review a previous result relating $IRES$-$UNSAT$ and $URES$-$UNSAT$, give upper bounds on the size of I-RES refutations, and prove a separation between the I-RES and U-RES proof systems.

A very important and useful previous result is the relationship between I-RES and U-RES proved by Chang:

Theorem 3.4 ([Cha70]). *A formula F has an* I-RES *refutation if and only if it has a* U-RES *refutation and therefore $IRES$-$UNSAT = URES$-$UNSAT$.*

This equivalence makes it easy to see that I-RES is an incomplete proof system, since there are unsatisfiable formulas which have no unit clauses, and therefore have no U-RES or I-RES refutations.

The following theorem proves upper bounds on the size of I-RES refutations:

Theorem 3.5. *For any formula $F \in IRES$-$UNSAT$ there exists an* I-RES *refutation of F with size at most $2n+1$, where n is the number of distinct variables in F.*

Proof: The proof is by induction on n:

Basis: For $n = 0$, $F = \{\emptyset\}$, which has an I-RES refutation containing $2n + 1 = 1$ clauses.

Induction Hypothesis: Suppose that our statement is true for $n - 1$.

Induction Step: We now show that our statement holds for n. Let F be any arbitrary formula in $IRES$-$UNSAT$ with n distinct variables. Since F has a U-RES refutation, it either contains a unit clause (x_i) or $(\neg x_i)$. Suppose that it contains the unit clause (x_i). Restrict F by setting x_i to True to produce the formula $F\restriction_{x_i=True}$, which has $n-1$ variables and also has an I-RES refutation. Our induction hypothesis therefore applies, so $F\restriction_{x_i=True}$ has an I-RES refutation π' of size $\leq 2(n-1)+1 = 2n-1$. Lifting the restriction on F and all corresponding clauses in π' yields an I-RES derivation π of the empty clause or the clause $(\neg x_i)$, where π has size exactly $2n - 1$. If π is a derivation of the empty clause, then we are done. In the case where π proves $(\neg x_i)$, recall that F contains the input clause (x_i), so simply resolve with this in order to produce the empty clause. In either case we have therefore produced an I-RES refutation containing $\leq 2n - 1 + 2 = 2n + 1$ clauses, as required. The case in which F contains the unit clause $(\neg x_i)$ is completely symmetrical: Simply restrict by $x_i = False$ instead, and then resolve with $(\neg x_i)$ at the end. □

However, although I-RES and U-RES are in some sense equivalent by Theorem 3.4 and I-RES has linear size upper bounds by Theorem 3.5, the same does not hold for U-RES proof size, and there exists a separation between the two proof systems. In fact, U-RES has $\Omega(n^2)$ size lower bounds on the following family of formulas:

Definition 3.6 **(F_U Formulas).** *There is one formula F_i in F_U for each value of $n \geq 1$, where n is the number of distinct variables. F_i is defined inductively as follows: $F_1 = \{\{x_1\}, \{\neg x_1\}\}$, and each subsequent F_i is created by taking F_{i-1}, adding the literal $\neg x_i$ to each clause, and finally adding the singleton clause $\{x_i\}$.*

A simple analysis of these formulas yields a separation between the I-RES and U-RES proof systems:

Theorem 3.7. *Each $F_n \in F_U$ requires* U-RES *refutations of size at least $\frac{n^2 + 3n + 2}{2}$, where n is the number of distinct variables in F.*

Proof: By construction, it is not hard to see that F_n contains exactly $n + 1$ clauses and that each variable x_i has exactly one positive occurrence, i negative occurrences, and deriving the empty clause requires all variables in the formula to be eliminated. Eliminating each x_i requires exactly i resolutions, although these resolution steps are only possible after the positive unit clause $\{x_i\}$ has been derived, since we are dealing with U-RES. F_n therefore requires $n + (n-1) + (n-2) + ... + 1$ applications of the resolution rule, which sums to exactly $\frac{n(n+1)}{2}$ clauses derived. In addition, each of the $n + 1$ initial clauses must be present in the proof. Any refutation of F_n therefore contains at least $\frac{n^2 + 3n + 2}{2}$ clauses, as required. $\square$

3.2 The Relationship Between Input Resolution and Horn Formulas

Horn formulas are an important class of formulas which come up rather frequently in computer science. For example, the algorithm employed by the Prolog programming language is called SLD Resolution, which is a special case of Horn Resolution. In this section we shall show that *HORN-UNSAT* is a proper subset of *IRES-UNSAT*:

Lemma 3.8. *HORN-UNSAT* $\subsetneq$ *IRES-UNSAT*

Proof: We first show that *HORN-UNSAT* $\subset$ *IRES-UNSAT* by proving that any unsatisfiable Horn formula F has an I-RES refutation. The proof is by induction on n:

Basis: For $n = 1$, F contains the clauses $\{x_1\}$ and $\{\neg x_1\}$, and therefore clearly has an I-RES refutation.

Induction Hypothesis: Assume that every Horn formula on n variables has an I-RES refutation.

Induction Step: Let F be any Horn formula on $n+1$ variables, and F' be any minimally unsatisfiable subset of its clauses. If F' contains fewer than $n+1$ variables, then we appeal to our Induction Hypothesis and are done, so we may assume that F' contains $n + 1$ variables. Any unsatisfiable formula contains at least one clause which has no positive literals in it, or else it could be satisfied by simply setting all variables to True. Similarly, it has at least one clause which has no negative literals in it, or else it could be satisfied by setting all variables to False. However, since F' is a Horn formula and it contains an all-positive clause, it must be a unit clause, so every unsatisfiable Horn formula contains a positive unit clause.

Let x_1 be the variable in a positive unit clause of F'. We restrict F' to create $F'\restriction_{x_1=True}$, which has only n variables, so our Induction Hypothesis applies, showing that it has an I-RES refutation π'. Since F' is minimally unsatisfiable, lifting the restriction on π' gives us π, which is an I-RES derivation of the clause $\{\neg x_1\}$. We resolve this with our positive unit clause $\{x_1\}$ to complete the refutation of F'. This is also a refutation of F, since we said that $F' \subset F$. Therefore by induction every Horn formula has an

I-RES refutation, so $HORN\text{-}UNSAT \subsetneq IRES\text{-}UNSAT$.

It is easy to see that the set inclusion is proper because there are many non-Horn formulas which have I-RES refutations. For instance, the unsatisfiable formula $F = (x \vee y) \wedge (\neg x) \wedge (\neg y)$ is not Horn but has an I-RES refutation. $\square$

This same argument immediately applies to the minimally unsatisfiable versions of these sets as well:

Corollary 3.9. $MU\text{-}HORN\text{-}UNSAT \subsetneq MU\text{-}IRES\text{-}UNSAT$

3.3 The Relationship Between Input Resolution and MU Formulas

Having established the relationship between $IRES\text{-}UNSAT$ and Horn formulas in the previous section, we now investigate the relationship between $IRES\text{-}UNSAT$ and $MU(k)$. In this section we will prove exact bounds on the size of I-RES proofs, and relate Horn Resolution, I-RES, and $MU(1)$. Finally, we will prove some facts about $MU(1)$, $MU\text{-}IRES\text{-}UNSAT$, and unit propagation which will be useful in later sections.

3.3.1 Exact Bounds on the Size of Input Resolution Refutations

The smallest k for which $MU(k)$ is interesting is $k = 1$. This is because a lower bound on the number of clauses in minimally unsatisfiable formulas is known, showing that the set $MU(0)$ is empty:

Lemma 3.10 ([AL86]). *Any minimally unsatisfiable CNF formula on n variables contains at least $n + 1$ clauses.*

This lower bound also has a useful corollary:

Corollary 3.11. *Any CNF formula with fewer than n clauses is not minimally unsatisfiable, and any unsatisfiable formula F containing exactly $n + 1$ clauses is minimally unsatisfiable (i.e. $F \in MU(1)$).*

In addition, Lemma 3.10 helps us to prove bounds on the size of refutations for minimally unsatisfiable formulas which hold for all forms of Resolution:

Corollary 3.12. *For any minimally unsatisfiable formula F, any refutation of F in any form of Resolution contains at least $2n + 1$ clauses.*

Proof: The lower bound is easy to see: Since F is minimally unsatisfiable, each of its clauses must be used in any refutation of F. Therefore by Lemma 3.10, any refutation of F contains at least $n+1$ clauses. However, since every variable in F is introduced into the proof at some point and must be eliminated, we need to derive at least n more clauses (one for each variable elimination), bringing our total to at least $2n + 1$. $\square$

Together with the upper bound from Theorem 3.5, this gives us tight bounds on the size of I-RES refutations for $MU\text{-}IRES\text{-}UNSAT$:

Corollary 3.13. *For any formula $F \in MU\text{-}IRES\text{-}UNSAT$ the size of every* I-RES *refutation of F is at least $2n + 1$, where n is the number of distinct variables in F. In addition, there exists an* I-RES *refutation of F with size at most $2n + 1$.*

Proof: The lower bound follows from Corollary 3.12, and the upper bound from Theorem 3.5. $\square$

3.3.2 Relating Horn Resolution, Input Resolution, and $MU(1)$

In [DDB98], Davydov, Davydova, and Kleine Büning prove a result corresponding to Lemma 3.10, except that theirs is specific to U-RES, and also includes an exact measure of the number of clauses for formulas in $MU\text{-}IRES\text{-}UNSAT$:

Theorem 3.14 ([DDB98]). *Any minimally unsatisfiable formula on n variables which has a* U-RES *refutation contains exactly $n + 1$ clauses.*

Together, Theorems 3.4 and 3.14 allow us to relate $MU\text{-}IRES\text{-}UNSAT$ to $MU(1)$:

Corollary 3.15. $MU\text{-}IRES\text{-}UNSAT \subsetneq MU(1)$

Proof: By Theorems 3.4 and 3.14, $MU\text{-}IRES\text{-}UNSAT \subset MU(1)$. It is easy to see that the set inclusion is proper because there exist formulas such as $F = (x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee z) \wedge (\neg x \vee \neg z)$ in $MU(1)$ which do not have I-RES refutations. □

Combining Lemma 3.8, Corollary 3.9, and Corollary 3.15 allows us to produce a simple set diagram which shows the exact relationship between $IRES\text{-}UNSAT$, $MU(1)$, $MU\text{-}IRES\text{-}UNSAT$, $HORN\text{-}UNSAT$, and $MU\text{-}HORN\text{-}UNSAT$. This diagram is shown below in Figure 1.

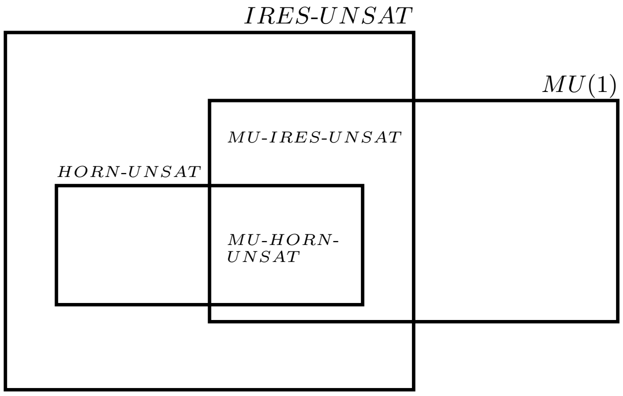

Figure 1: The Relationship Between I-RES, Horn Formulas, and MU(1) Formulas

3.3.3 MU(1), MU-IRES-UNSAT, and Unit Propagation

We now prove that both $MU(1)$ and $MU\text{-}IRES\text{-}UNSAT$ are closed under unit propagation, a fact which will be useful in subsequent sections.

Lemma 3.16. *For any CNF formula F, if F contains two clauses C_1 and C_2 such that $C_1 \subset C_2$, then F is not minimally unsatisfiable.*

Proof: Let C_1 and C_2 be clauses of F such that $C_1 \subset C_2$. If F is satisfiable, then it is not minimally unsatisfiable, and we are done, so assume that it is unsatisfiable. Let $F' = F - \{C_2\}$. Because $C_1 \subset C_2$, if F' is satisfiable, then the same truth assignment which satisfies it also satisfies F, so F' is not satisfiable. But F' contains a proper subset of the clauses of F, therefore showing that F is not minimally unsatisfiable. □

We can use this lemma to prove that $MU(1)$ is closed under unit propagation:

Lemma 3.17. *For any formula $F \in MU(1)$, if F contains a unit clause $\{l\}$ for some literal l, then $F\lceil_{l=True} \in MU(1)$.*

Proof: If $F \in MU(1)$ contains a unit clause $\{l\}$ for some literal l, then l does not occur anywhere else in F by Lemma 3.16. Therefore $F\lceil_{l=True}$ contains $n-1$ variables and exactly n clauses, which means that it is minimally unsatisfiable by Corollary 3.11, so $F\lceil_{l=True} \in MU(1)$, as required. □

Combining the previous lemma with Corollary 3.15 yields the following corollary which shows that $MU\text{-}IRES\text{-}UNSAT$ is closed under unit propagation:

Corollary 3.18. *For any formula $F \in MU\text{-}IRES\text{-}UNSAT$, if F contains a unit clause $\{l\}$ for some literal l, then $F\lceil_{l=True} \in MU\text{-}IRES\text{-}UNSAT$.*

3.4 A Matrix Characterization of MU-IRES-UNSAT

Although matrix algebra and propositional formulas may at first glance have very little to do with each other, matrices can be used to encode formulas. This can be done by giving each cell in the matrix one of three values: $+, -$, and 0 such that each row in the matrix represents a distinct variable, and each column represents a distinct clause. If column j of row i is $+$, then the literal x_i occurs in clause j. Similarly, if column j in row i is $-$, then the literal $\neg x_i$ occurs in clause j. Finally, if position (i, j) in the matrix is 0, then the variable i does not occur in clause j.

For example, the formula $F = (x_1) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3)$ has the following matrix representation:

$$\begin{pmatrix} + & - & 0 \\ 0 & + & - \\ 0 & 0 & + \end{pmatrix}$$

In [DDB98] the authors show that the 'basic matrices', which are of interest in the area of linear algebra exactly capture the formulas in $MU(1)$. From Corollary 3.15, we know that $MU\text{-}IRES\text{-}UNSAT$ is a proper subset of $MU(1)$, so it stands to reason that $MU\text{-}IRES\text{-}UNSAT$ comprises a proper subset of the basic matrices. In this section we shall show that this is in fact the case. We begin by defining this subest of matrices, which we call the 'sub-basic matrices':

Definition 3.19 (Sub-Basic Matrices). *The sub-basic matrices are defined inductively:*

1. $\begin{pmatrix} + & - \end{pmatrix}$ *is a sub-basic matrix.*
2. *If B is a sub-basic matrix and b is a vector with $b_j \in \{-, 0\}$ with at least one $-$-sign, then the following matrix is also sub-basic:*

$$\begin{pmatrix} + & b \\ 0 & B \end{pmatrix}$$

3. *If B is a sub-basic matrix and b is a vector with $b_j \in \{+, 0\}$ with at least one $+$-sign, then the following matrix is also sub-basic:*

$$\begin{pmatrix} - & b \\ 0 & B \end{pmatrix}$$

Having defined the sub-basic matrices, we now show that a formula is in MU-$IRES$-$UNSAT$ if and only if it can be encoded by a sub-basic matrix. We begin by proving the forward direction:

Lemma 3.20. *Any formula $F \in MU$-$IRES$-$UNSAT$ can be represented as a sub-basic matrix.*

Proof: The proof is by induction on n, the number of distinct variables in F:

Basis: For $n = 1$, $F = (x_1) \wedge (\neg x_1)$, and $\begin{pmatrix} + & - \end{pmatrix}$ is sub-basic, so our statement holds for the base case.

Induction Hypothesis: Suppose that any formula $F \in MU$-$IRES$-$UNSAT$ containing n variables can be represented as a sub-basic matrix.

Induction Step: Let F be any arbitrary formula in MU-$IRES$-$UNSAT$ containing $n+1$ variables. Since $F \in MU$-$IRES$-$UNSAT$, it contains a unit clause. Without loss of generality, let us call this unit clause $\{l_1\}$, where $l_1 = x_1$ or $\neg x_1$. Let F' be $F\lceil_{l_1=True}$, which has n variables and by Corollary 3.18 is in MU-$IRES$-$UNSAT$. Therefore our induction hypothesis applies, and F' has the sub-basic matrix B. By Lemma 3.16 l_1 does not appear in any other clause of F, so if $l_1 = x_1$, then F has the sub-basic matrix $\begin{pmatrix} + & b \\ 0 & B \end{pmatrix}$, where b is a vector with $b_j \in \{-, 0\}$ with at least one $-$-sign (because F is minimally unsatisfiable). The case where $l_1 = \neg x_1$ is completely analogous, so in either case F has a sub-basic matrix, and by induction any formula $F \in MU$-$IRES$-$UNSAT$ can be represented as a sub-basic matrix. $\square$

We now prove the reverse direction:

Lemma 3.21. *Every sub-basic matrix encodes a formula $F \in MU$-$IRES$-$UNSAT$.*

Proof: The proof is by induction on n, the number of rows of a sub-basic matrix:

Basis: For $n = 1$, the only sub-basic matrix is $\begin{pmatrix} + & - \end{pmatrix}$, corresponding to $F = (x_1) \wedge (\neg x_1)$, which is in MU-$IRES$-$UNSAT$.

Induction Hypothesis: Suppose that every sub-basic matrix containing n rows encodes a formula $F \in MU$-$IRES$-$UNSAT$.

Induction Step: Let M by any arbitrary sub-basic matrix containing $n+1$ rows. By definition, M is $\begin{pmatrix} + & b \\ 0 & B \end{pmatrix}$, where b is a vector with $b_j \in \{-, 0\}$ with at least one $-$-sign, or M is $\begin{pmatrix} - & b \\ 0 & B \end{pmatrix}$, where b is a vector with $b_j \in \{+, 0\}$ with at least one $+$-sign. In either case, B is sub-basic and contains n variables, so our induction hypothesis applies, and it encodes a formula $F_B \in MU$-$IRES$-$UNSAT$. In our first case, we add the singleton clause $\{x_1\}$ to F_B and add the literal $\neg x_1$ to at least one clause of F_B to produce F, which is clearly still unsatisfiable and has a U-RES refutation, so therefore also has an I-RES refutation by Theorem 3.4. However, F contains $n+1$ variables and $n+2$ clauses, so by Corollary 3.11 it is minimally unsatisfiable.

Our second case where we add the singleton clause $\{\neg x_1\}$ to F_B and add the literal x_1 to at least one clause of F_B to produce F is completely analogous. Therefore, in either case M encodes a formula $F \in MU$-$IRES$-$UNSAT$, so by induction every sub-basic matrix encodes a formula $F \in MU$-$IRES$-$UNSAT$. $\square$

Putting these two lemmas together shows that the sub-basic matrices are a perfect characterization of MU-$IRES$-$UNSAT$:

Theorem 3.22. *A formula F is in MU-$IRES$-$UNSAT$ if and only if it can be encoded as a sub-basic matrix.*

This result of course has implications for $IRES$-$UNSAT$ as well, because it shows that any minimally unsatisfiable subset of clauses from formula which has an I-RES refutation can be encoded as a sub-basic matrix. In other words, any formula $F \in IRES$-$UNSAT$ can be encoded as a sub basic matrix which has been augmented with any arbitrary clauses, giving us a matrix characterization of $IRES$-$UNSAT$ as well.

4 Tractable Aspects of Input Resolution

In the previous section we related various languages such as $HORN$-$UNSAT$, $MU(1)$, and MU-$IRES$-$UNSAT$ to $IRES$-$UNSAT$. We shall use these results to explore the complexities of some of the tractable aspects of the I-RES proof system. First we shall review some previous results showing that $HORN$-$UNSAT$ and $IRES$-$UNSAT$ are $\mathcal{P}$-Complete, and that $MU(1) \in \mathcal{P}$. Following this we show that MU-$IRES$-$UNSAT \in \mathcal{P}$ and that the size problem for MU-$IRES$-$UNSAT$ is in $\mathcal{P}$.

4.1 The Complexities of HORN-UNSAT, IRES-UNSAT, and MU(1)

We now review some previous results about the tractability of $HORN$-$UNSAT$, $IRES$-$UNSAT$, and $MU(1)$. For example, the complexity of $HORN$-$UNSAT$ is well-understood:

Theorem 4.1 ([Pla84], [Pap94], p.176). *The problem of determining whether or not a given Horn formula is satisfiable (i.e. of deciding the language $HORN$-SAT) is $\mathcal{P}$-Complete.*

This shows that determining whether or not Horn formulas are unsatisfiable is co$\mathcal{P}$-Complete, so since $\mathcal{P}$ is closed under complement, this immediately gives us the complexity of $HORN$-$UNSAT$:

Corollary 4.2. *$HORN$-$UNSAT$ is $\mathcal{P}$-Complete.*

Similar $\mathcal{P}$-Completeness results have been proved for $URES$-$UNSAT$:

Theorem 4.3 ([JL77]). *The problem of determining whether or not a given formula has a* U-RES *refutation (i.e. of deciding the language $URES$-$UNSAT$) is $\mathcal{P}$-Complete.*

Since we know from Theorem 3.4 that $IRES$-$UNSAT = URES$-$UNSAT$, this immediately implies the $\mathcal{P}$-Completeness of $IRES$-$UNSAT$:

Corollary 4.4. *$IRES$-$UNSAT$ is $\mathcal{P}$-Complete, and furthermore there exists an $O(n \cdot m)$ algorithm which takes as input a formula F and determines whether or not it has an* U-RES *(and therefore* I-RES*) refutation, where n is the number of distinct variables in F, and m is the number of clauses.*

Proof: Showing that $IRES$-$UNSAT$ is $\mathcal{P}$-Complete is trivial, since $IRES$-$UNSAT = URES$-$UNSAT$, which is $\mathcal{P}$-Complete by Theorem 4.3. It can be solved by the following unit propagation algorithm: Repeatedly pick a unit clause and resolve it with every other clause possible. Since U-RES is closed under restriction, this algorithm is clearly correct, will take at most $O(n \cdot m)$ time in total, and will either yield the empty clause, in which case it has a U-RES refutation (and therefore I-RES refutation by Theorem 3.4), or a formula where no more unit resolutions are possible, in which case it does not.

In addition, the complexity of $MU(k)$ has been settled for $k = 1$:

Theorem 4.5 ([DDB98]). *$MU(1) \in \mathcal{P}$ and there exists an $O(n^2)$ algorithm which decides if any arbitrary formula F is in $MU(1)$, where n is the number of distinct variables in F.*

The complexity of $MU(k)$ for all $k \geq 2$ remains open.

4.2 The Tractability of MU-IRES-UNSAT

Having proved that $IRES\text{-}UNSAT$ is $\mathcal{P}$-Complete, we now show that $MU\text{-}IRES\text{-}UNSAT \in \mathcal{P}$ by counting clauses:

Corollary 4.6. *$MU\text{-}IRES\text{-}UNSAT \in \mathcal{P}$ and has an $O(n \cdot m)$ algorithm, where n is the number of distinct variables and m is the number of clauses.*

Proof: Given a formula F, we determine if $F \in MU\text{-}IRES\text{-}UNSAT$. By Corollary 3.11 we know that any formula containing fewer than n variables is not minimally unsatisfiable, so if this is the case, then reject. Similarly, by Corollary 3.15, any formula containing more than $n+1$ variables cannot be in $MU\text{-}IRES\text{-}UNSAT$, so if this is the case, then reject. Therefore we are left with the case where F contains exactly $n+1$ clauses, so it is either has an I-RES refutation (in which case it is in $MU\text{-}IRES\text{-}UNSAT$) or it does not (in which case it is not). This can be determined using the $O(n \cdot m)$ algorithm from Corollary 4.4. □

4.3 The Tractability of the MU-IRES-UNSAT Size Problem

The size problem for $MU\text{-}IRES\text{-}UNSAT$ takes as input a formula / integer pair (F, k) and asks if F is minimally unsatisfiable and has an I-RES refutation of size at most k. We now show that this problem is in $\mathcal{P}$:

Corollary 4.7. *Given a formula F and integer k, the problem of determining whether or not $F \in MU\text{-}IRES\text{-}UNSAT$ and has an* I-RES *refutation of size at most k is in $\mathcal{P}$ and has an $O(n \cdot m)$ algorithm, where n is the number of distinct variables in F, and m is the number of clauses.*

Proof: Simply use the algorithm from Corollary 4.6 to determine if $F \in MU\text{-}IRES\text{-}UNSAT$. If not, then reject, and otherwise compare k with $2n + 1$. If $k \geq 2n + 1$, then by Corollary 3.13 we can simply accept, and otherwise we reject. □

4.4 The Tractability of the MU-IRES-UNSAT Problem with Top Clause

In previous sections we have seen that various versions of the $MU\text{-}IRES\text{-}UNSAT$ problem are in $\mathcal{P}$. We now prove that another generalized form of the problem is also tractable. Instead of asking whether a minimally unsatisfiable formula F has an I-RES refutation, we can ask if it has an I-RES refutation in which one of the top clauses is C. We shall refer to this as the $MU\text{-}IRES\text{-}UNSAT$ problem with top clause. In order to prove that it is in $\mathcal{P}$, we will first prove a 'top clause' lemma:

Lemma 4.8 (Top Clause Lemma). *If a formula $F \in MU(1)$ has an* I-RES *refutation, then it has an* I-RES *refutation with any arbitrary clause $C \in F$ as one of the top clauses.*

Proof: By induction on n, the number of distinct variables in F:

Basis: In order to simplify later arguments, we will cover the cases of $n = 0$ and $n = 1$ as our base cases. For $n = 0$, $F = \emptyset$, which trivially unsatisfiable, and its only proof contains only the empty clause. For $n = 1$, the only minimally unsatisfiable formula is $F = (x) \wedge (\neg x)$, in which the only refutation has both of its clauses as top clauses, so our statement holds for both $n = 0$ and $n = 1$.

Induction Hypothesis: Suppose that our statement is true for $n-1$.

Induction Step: We now show that our statement holds for n. Let F be any arbitrary formula in $MU(1)$ on n variables which has an I-RES refutation. By Theorem 3.4, F must contain a unit clause, call it $\{l\}$ for some literal l. We want to show that F has an I-RES refutation with top clause C, where C is any arbitrary clause in F. In order to apply our induction hypothesis, we restrict F to produce $F' = F\lceil_{l=True}$. By Lemma 3.17, $F\lceil_{l=True} \in MU(1)$. In addition, I-RES is closed under restriction, so our induction hypothesis applies to F', which therefore has an I-RES refutation with any arbitrary $C\lceil_{x=True} \in F'$ as top clause; let us call this proof π'.

There are two cases to consider:

Case 1: Suppose that $C \neq \{l\}$. Since $F \in MU(1)$ and $\{l\} \in F$ is a unit clause, Lemma 3.16 applies, so the literal l does not occur anywhere else in F, including C. Similarly, $C \neq \{\neg l\}$, since this is not the base case. Therefore the restricted clause $C\lceil_{l=True}$ does not disappear from F'. Furthermore, since F' is minimally unsatisfiable, every clause is used in π', so lifting the restriction of $l = True$ yields π which is an I-RES derivation from F of $\{\neg l\}$ with top clause C. We resolve the final clause $\{\neg l\}$ of π with our unit input clause $\{l\}$ to complete the refutation.

Case 2: Suppose that $C = \{l\}$. We know that F is minimally unsatisfiable, so it must contain a clause of the form $D = E \cup \{\neg l\}$, and $n \geq 2$, so $E \in F'$. By our induction hypothesis, there exists an I-RES refutation π' from F' with top clause E. Lift the restriction of $l = True$ from all clauses of π', but don't add $\{\neg l\}$ back to the top clause E. Now resolve $\{l\}$ with $D = E \cup \{\neg x\}$ at the top of π' to produce E, and if necessary, also resolve $\{l\}$ with the singleton clause $\{\neg l\}$ (if it exists) at the bottom of the proof to complete the refutation. This yields an I-RES refutation from F with top clause $C = \{l\}$.

Therefore, in either case we have produced an I-RES refutation from F with top clause C, as required, and our result follows by induction. □

This lemma allows us to prove that the *MU-IRES-UNSAT* problem with top clause is in $\mathcal{P}$:

Theorem 4.9. *Given a minimally unsatisfiable formula F and a clause $C \in F$, determining if F has an* I-RES *refutation with top clause C is in $\mathcal{P}$.*

Proof: To show that this problem is in $\mathcal{P}$, we first determine if $F \in$ *MU-IRES-UNSAT*, which can be done in polynomial time by Corollary 4.6. If not, then reject. Otherwise, by Lemma 4.8 F has an I-RES refutation with C as top clause, so accept. □

5 Automatizability of Input Resolution

In this section we show that I-RES is automatizable and that *MU-IRES-UNSAT* is optimally automatizable. These results form an interesting contrast with the results in the next section which show that the problem of computing the optimum size of I-RES refutations is $\mathcal{NP}$-Complete.

Informally, a proof system S is automatizable if there exists an algorithm which can always find proofs which are only polynomially larger than the optimal. Such algorithms are obviously of great practical interest because they allow us to efficiently automate theorem proving. More formally, automatizability is defined as follows:

Definition 5.1 (Automatizability). *A propositional proof system S is automatizable if there exists a deterministic algorithm A such that for every formula $F \in UNSAT$ (or $TAUT$), A returns an S-proof of F in time polynomial in $|F|$ and $|\pi|$, where $|F|$ is the number of clauses in F, and $|\pi|$ is the size of the smallest S-proof of F.*

Because they form the basis of so many practical SAT-solving algorithms, RES and its refinement T-RES are some of the best candidate proof systems which researchers would like to automatize. However, under widely-believed cryptographic assumptions, they are not automatizable [AR02].

It is therefore likely that the only possible automatizable Resolution refinements are incomplete. In other words, results such as the ones in this section are in a sense the best we can hope for.

The most obvious candidate for automatizability is Horn Resolution. Our results from previous sections make it easy to see that Horn formulas are an automatizable example of restricted SAT inputs:

Corollary 5.2. I-RES *on* $HORN\text{-}UNSAT$ *is automatizable and there exists an* $O(n \cdot m)$ *algorithm which takes as input any Horn formula* F *(where* n *is the number of distinct variables in* F *and* m *is the number of clauses), and either produces an* I-RES *refutation containing at most* $2n + 1$ *clauses, or returns that no refutation exists.*

Proof: We simply use the $O(n \cdot m)$ algorithm from Corollary 4.4 to determine if F is unsatisfiable. If not, then return that no refutation exists. Otherwise use the $O(n \cdot m)$ algorithm implicit in Lemma 3.8 to produce the desired I-RES refutation containing at most $2n + 1$ clauses. □

Another restriction which has received much attention is 2-SAT, in which we require the input formula to have exactly two literals per clause. Unlike Horn-SAT, which is $\mathcal{P}$-Complete, 2-SAT is $\mathcal{NL}$-Complete [Pap94, p.398]. In addition, 2-SAT is also known to be automatizable [Coo71], and in fact optimal polytime automatizability algorithms exist for finding the shortest T-RES refutation [Sub04] and the shortest RES refutation of any unsatisfiable 2-CNF formula [BOM07].

We now prove that for I-RES there exists a polytime algorithm for finding a refutation which is at worst linearly longer than the optimal, thereby showing its automatizability. In fact, I-RES is strongly automatizable in the sense that we can always find a refutation which is polynomial in the length of the input formula. Automatizability with respect to formula size is usually not even a reasonable thing to ask for because many proof systems have exponential size lower bounds, making this impossible. This further shows just how tractable the I-RES proof system is.

Theorem 5.3. *The* I-RES *proof system is automatizable. More specifically, given a formula* F *containing* n *distinct variables and* m *clauses, there exists an* $O(n \cdot m)$ *algorithm which finds an* I-RES *refutation containing at most* $2n + 1$ *clauses or reports that* F *has no* I-RES *refutation.*

Proof: First use the $O(n \cdot m)$ algorithm from Corollary 4.4 to determine if F has an I-RES refutation. If not, then we report that none exists. Next we apply the $O(n \cdot m)$ DPLL algorithm implicit in the induction step of Theorem 3.5 to produce an I-RES refutation of F containing at most $2n+1$ clauses. □

It is worth noting that for minimally unsatisfiable formulas this algorithm produces the shortest possible proof, but if F is not minimally unsatisfiable, then this algorithm may not produce the optimal.

6 The $\mathcal{NP}$-Completeness of Calculating I-RES Proof Size

In previous sections we saw that I-RES is automatizable, $IRES\text{-}UNSAT$ is $\mathcal{P}$-Complete, that I-RES is optimally automatizable on $MU\text{-}IRES\text{-}UNSAT$, and the size problem for $MU\text{-}IRES\text{-}UNSAT$ is in $\mathcal{P}$. However, in this section we show an interesting contrast with these tractability results by noting that the size problem for $IRES\text{-}UNSAT$ is $\mathcal{NP}$-Complete. This result follows immediately from the main result of Alekhnovich, Buss, Moran and Pitassi [ABMP01]:

Theorem 6.1 ([ABMP01]). *The problem of approximating the size (regardless of whether size is measured in total symbols or number of clauses) of the smallest Resolution refutations of formulas from* $HORN\text{-}UNSAT$ *to within a factor of* $2^{log^{1-o(1)} n}$ *is* $\mathcal{NP}$*-Hard.*

Corollary 6.2. *The problem of approximating the size (regardless of whether size is measured in total symbols or number of clauses) of the smallest* I-RES *or* U-RES *proofs of formulas from $IRES\text{-}UNSAT$ to within a factor of $2^{log^{1-o(1)}n}$ is $\mathcal{NP}$-Hard.*

Proof: There is no need to change the proof in [ABMP01]; it is easy to see that the reduction from Circuit MMSA produces Horn formulas which have both I-RES and U-RES refutations, so the result holds for these proof systems as well. □

This corollary in turn yields another one, namely that given a formula F and integer k, the problem of determining if F has an I-RES refutation of size at most k is $\mathcal{NP}$-Complete. The language associated with this problem is defined as follows:

Definition 6.3 ($IRES\text{-}SIZE$)**.** $IRES\text{-}SIZE = \{(F, k) \mid F$ *is a formula for which there exists a* I-RES *refutation with size at most* $k\}$

Corollary 6.4. *$IRES\text{-}SIZE$ is $\mathcal{NP}$-Complete under Turing reducibility.*

Proof: We first show that $IRES\text{-}SIZE$ is in $\mathcal{NP}$: $IRES\text{-}UNSAT$ is in $\mathcal{P}$ by Corollary 4.4 and is therefore also in $\mathcal{NP}$, so we first check to see if F has an I-RES refutation. If it does not, then we reject. Otherwise it has an I-RES refutation of size at most $2n + 1$ by Theorem 5.3. We therefore compare k with $2n + 1$ and if k is greater, then we accept immediately. Otherwise we nondeterministically guess the shortest I-RES refutation of F, which we know must have a size of at most $2n + 1$ and accept if and only if its size is at most k, thereby completing our $\mathcal{NP}$ algorithm.

Next we show that $IRES\text{-}SIZE$ is $\mathcal{NP}$-Hard via a Turing reduction from the problem of approximating minimum I-RES refutation size which was proved $\mathcal{NP}$-hard in Corollary 6.2. Assuming that we have an algorithm for $IRES\text{-}SIZE$, we can use it as a subroutine to build an algorithm for determining the size of the smallest refutation of F as follows: By Theorem 5.3, F has an I-RES refutation containing at most $2n + 1$ clauses. We therefore perform a binary search between the range 1 and $2n + 1$ using our $IRES\text{-}SIZE$ algorithm to determine the size of the smallest refutation. Since this is only a logarithmic number of subroutine calls, it follows that $IRES\text{-}SIZE$ is $\mathcal{NP}$-Complete, as required. □

7 Previous Results Related to Resolution Space

7.1 Pebbling

An important survey on pebbling by Pippenger [Pip80] is a very good overview of the area. The literature also contains some useful results for manipulating pebbling strategies. The first is a connection between normal pebbling and pebbling with sliding. In [GLT80], Gilbert, Lengauer, and Tarjan prove that any DAG G can be black pebbled using k pebbles with sliding if and only if it can be black-pebbled using $k + 1$ pebbles without sliding. We generalize this result to the black-white pebbling of monotone circuits:

Lemma 7.1. *Any monotone circuit C can be black-white pebbled using k pebbles with sliding if and only if it can be black-white pebbled using $k + 1$ pebbles without sliding.*

Proof: ⇒ Suppose that C can be black-white pebbled using k pebbles with sliding. Replace each instance of the sliding rule from gate x to gate y with two moves; first place a pebble of the same colour as the pebble on x onto y, and then remove the pebble from x. This substitution increases the number of pebbles used by 1 momentarily during the intermediate step and works regardless of whether the pebble is white or black, regardless of what type of gate x is, and leaves us with a pebbling strategy that involves no sliding and requires at most $k + 1$ pebbles, as required.

⇐ Suppose that C can be black-white pebbled using $k + 1$ pebbles without sliding, and let H be the pebbling strategy associated with this pebbling.

Let (B_i, w_i) be any time in H containing $k+1$ pebbles. Therefore, the move transitioning from step (B_{i-1}, w_{i-1}) to (B_i, w_i) was the placement of either a black or white pebble on a vertex x. Either this is the final move in the pebbling or it is not. If this is the final move, then it must be the placement of a black pebble. Then all of x's predecessors must be pebbled at step (B_{i-1}, w_{i-1}), and there are no white pebbles on C, so simply slide one of the black pebbles from one of x's predecessors to x instead of placing a new black pebble on x, thereby saving a pebble.

If this is not the final move in the pebbling, then the pebble being placed on x is either black or white, and the move transitioning from step (B_i, w_i) to step (B_{i+1}, w_{i+1}) must be the removal of a pebble from a vertex y. If the pebble being placed on x is white, then it does not require any predecessors, so simply remove the pebble from y before placing the white pebble on x, thereby saving a pebble.

If the pebble being placed on x is black, then y either is or is not an immediate predecessor of x. If y is not an immediate predecessor, then it is not needed for the placement of x, so simply remove the pebble from y first, and then place the black pebble on x, thereby saving a pebble.

If y is an immediate predecessor of x, then the pebble being removed from y is either black or white. If it is black, then instead of placing a black pebble on x and then removing the black pebble from y, we slide the black pebble from y to x, thereby saving a pebble and still ending up in the same pebbling configuration (B_{i+1}, w_{i+1}).

If the pebble being removed from y is white, then instead of placing a black pebble on x and removing a white pebble from y, we first remove the white pebble from y, then place a black pebble on y, and then slide it from y to x, once again saving a pebble and ending up in the same pebbling configuration (B_{i+1}, w_{i+1}).

We perform this local transformation to save a pebble at every stage of H where $k+1$ pebbles are used. This shows that C can be pebbled using k pebbles with sliding, as required. $\square$

Another useful result for manipulating pebblings is found in [Hei81]. It it, Meyer Auf Der Heide proves that for any DAG G, any black-white pebbling strategy (without sliding) using k pebbles can be turned into its 'dual' white-black pebbling strategy using k pebbles by reversing the strategy, converting all black pebbles to white and all white pebbles to black.

We generalize this result to black-white pebbling monotone circuits with the sliding rule present:

Lemma 7.2. *Given a black-white pebbling strategy (with sliding) of a monotone circuit C using at most k pebbles, if one reverses the strategy, converts all black pebbles to white, and all white pebbles to black, then the resulting strategy is also a valid black-white pebbling strategy of C which uses at most k pebbles.*

Proof: From the description of the black-white pebbling game in Definition 2.5, it is easy to see that every possible move in the game has a corresponding countermove which is exactly equivalent to making the move in reverse with all pebbles having exactly the opposite colour. For example, consider a node v in C with $d \geq 0$ predecessors $u_1, ..., u_d$. The following are pebbling moves together with their duals:

1. Let v be an unpebbled AND gate with all predecessors pebbled, and suppose we place a black pebble on v. In this case the dual move is to start with a white pebble on v, have all other corresponding nodes contain opposite pebbles, and then remove the white pebble from v.

2. Let v be an unpebbled OR gate with at least one predecessor pebbled, and suppose we place a black pebble on v. In this case the dual move is to start with a white pebble on v, have all other corresponding nodes contain opposite pebbles, and then remove the white pebble from v.

3. Let v be an unpebbled AND gate with all predecessors pebbled and suppose that we slide a black pebble from some predecessor u_i to v. In this case the dual move is to start with a white pebble on v but none on u_i, have all other corresponding nodes contain opposite pebbles, and then slide the white pebble from v to u_i.

4. Let v be an unpebbled OR gate with at least one predecessor pebbled and suppose that we slide a black pebble from some predecessor u_i to v. In this case the dual move is to start with a white

pebble on v but none on u_i, have all other corresponding nodes contain opposite pebbles, and then slide the white pebble from v to u_i.

5. Suppose that a black pebble is removed from v, where v is either type of gate. In this case the dual move is to have all other corresponding nodes pebbled with opposite pebbles, and then place a white pebble on v.

Of course, all of these dual relationships hold in the opposite direction as well, and each of these cases the number of total pebbles used is identical. It is therefore possible to take any black-white pebbling strategy (with sliding) and convert it step-by-step into a dual strategy which uses exactly the same number of pebbles, as required. □

7.2 Space & Games

The pebbling game is related closely to RES clause space. Let F be any arbitrary unsatisfiable formula, let π be a configuration-style RES refutation of F, and let G be the DAG underlying the structure of π. It is not hard to see that the clause space used in computing π is exactly equal to $B\text{-}Peb(G)$. More specifically, $CS(F \vdash_{\mathsf{RES}} \emptyset)$ is equal to the pebbling number of the DAG with the smallest pebbling number of all DAGs underlying valid RES refutations of F. This shows that both general clause space as well as tree clause space are related closely to the pebbling game.

A particularly relevant result relating black-white pebbling number to space was proved by Ben-Sasson:

Theorem 7.3 ([BS02]). *For any monotone circuit C, $BW\text{-}Peb(C) \leq TS(Peb^1(C) \vdash_{\mathsf{RES}} \emptyset)$, where $BW\text{-}Peb(C)$ is defined without sliding.*

Implicit in the proof of this Theorem is the following Corollary:

Corollary 7.4 ([BS02]). *For any monotone circuit C, $BW\text{-}Peb(C) \leq VS(Peb^1(C) \vdash_{\mathsf{RES}} \emptyset)$, where $BW\text{-}Peb(C)$ is defined without sliding.*

Tree clause space is also related closely to the Prover/Delayer game, a two-player combinatorial game played on an unsatisfiable formula. The results in [ET03] are the definitive work relating the Prover/Delayer number $PD(F)$ of an unsatisfiable CNF formula to its tree clause space $TCS(F \vdash_{\mathsf{T\text{-}RES}} \emptyset)$ by showing the following:

Theorem 7.5 ([ET03]). *For any unsatisfiable CNF formula F, $TCS(F \vdash_{\mathsf{T\text{-}RES}} \emptyset) = PD(F) + 1$.*

In other words, the Prover/Delayer game perfectly captures the notion of clause space for T-RES.

In addition to doing some of the pioneering research in the area of Resolution space, Esteban & Torán also showed that an upper bound on tree clause space gives an upper bound on the size of T-RES proofs:

Theorem 7.6 ([ET01]). *Let F be an unsatisfiable formula on n distinct variables. If F has a* T-RES *refutation with tree clause space $s = TCS(F \vdash_{\mathsf{T\text{-}RES}} \emptyset)$, then it has a* T-RES *refutation of size $\binom{n+s}{s}$.*

The Prover/Delayer game can also be used to give a lower bound on T-RES size; in [BSIW04], the authors prove that lower bounds on $PD(F)$ can be used to prove lower bounds on the size of T-RES proofs:

Theorem 7.7 ([BSIW04]). *If the Delayer has a strategy guaranteed to win $> k$ points on F, then every* T-RES *refutation of F has size $> 2^k$.*

In other words, upper and lower bounds on the Prover/Delayer Number of a formula not only immediately imply corresponding upper and lower bounds on tree clause space (since Prover/Delayer Number and tree clause space are basically synonymous), but they also imply upper and lower bounds for T-RES proof size.

7.3 Space & Width

An important paper relating space and width is [AD03]. In it the authors prove that for unsatisfiable k-CNF formulas, space is an upper bound on width.

Theorem 7.8 ([AD03]). *Let F be any unsatisfiable CNF formula. Then*

$$CS(F \vdash_{\mathsf{RES}} \emptyset) \geq w(F \vdash_{\mathsf{RES}} \emptyset) - w(F).$$

This is a very powerful result, because it can be used to derive space lower bounds for all formulas for which width lower bounds are known.

7.4 Width & Size

The relationship between width & size is very important because it shows that lower bounds for width imply lower bounds for size. This simplifies proofs for size lower bounds by allowing us to focus on lower bounds for width. The main result linking width and size is [BSW01], and is also explained well in [Urq06]:

Theorem 7.9 ([BSW01]). *Let F be a contradictory set of clauses with an underlying set of variables V, and let $S(F \vdash_{\mathsf{RES}} \emptyset)$ be the minimum size of any* RES *refutation of F. Then*

$$S(F \vdash_{\mathsf{RES}} \emptyset) = exp\left(\Omega\left(\frac{(w(F \vdash_{\mathsf{RES}} \emptyset) - w(F))^2}{|V|}\right)\right)$$

7.5 Tradeoff Results

Another important paper is [BS02], in which the author proves a number of interesting tradeoff results. For example, he provides families of formulas for which there are:

- linear size T-RES proofs and constant width T-RES proofs, but no T-RES proof that has both small width and small size.
- RES proofs with constant clause space, and RES proofs with constant width, but no RES proof that has both small width and small clause space.
- linear size RES proofs that also have constant width, but no RES proof that has both small clause space and small size.

These results rely on the pebbling contradiction formulas from Definition 2.7 above.

A more recent paper in the same vein is [Nor06]. In it, the author proves a tradeoff result for Resolution width and clause space. More specifically, he proves the existence of a family of unsatisfiable formulas with constant width but clause space which is bounded by $\Theta(lg(n))$, where n is the formula length.

7.6 Complexity of Pebbling

A number of complexity results involving pebbling are known. In [Lin78], the author proves the following interesting result about the black pebbling game on monotone circuits:

Theorem 7.10 ([Lin78]). *Given a monotone circuit C and an integer k, the problem of determining if C can be black-pebbled with k pebbles (with sliding) is $\mathcal{PSPACE}$-Complete.*

This result was later extended to DAGs in [GLT80]:

Theorem 7.11 ([GLT80]). *Given a DAG G and an integer k, the problem of determining if G can be black-pebbled with k pebbles (with sliding) is $\mathcal{PSPACE}$-Complete.*

More recently, the black-white pebbling game on monotone circuits was also shown to be $\mathcal{PSPACE}$-Complete [HP07]:

Theorem 7.12 ([HP07]). *Given a monotone circuit C and an integer k, the problem of determining if C can be black-white-pebbled with k pebbles (with sliding) is $\mathcal{PSPACE}$-Complete.*

This was also extended to DAGs in [HP07], thereby settling the long-standing open problem of settling its complexity:

Theorem 7.13 ([HP07]). *Given a DAG G and an integer k, the problem of determining if G can be black-white-pebbled with k pebbles (with sliding) is $\mathcal{PSPACE}$-Complete.*

By Lemma 7.1, we get the following Corollary:

Corollary 7.14. *The $\mathcal{PSPACE}$-Completeness of all four versions of the pebbling game above holds regardless of whether or not sliding is allowed.*

All of these results are particularly interesting because they are $\mathcal{PSPACE}$-Completeness results for a game which has only one player, whereas most $\mathcal{PSPACE}$-Complete games have two.

One final result worth mentioning is [HU07] in which we reduced from black pebbling monotone circuits to prove that the problems of calculating T-RES space requirements and determining the winner of the Prover/Delayer game are both $\mathcal{PSPACE}$-Complete.

8 The Complexity of Input Resolution Derivation Total Space

This section contains a number of results pertaining to black pebbling and the total space of I-RES. In Section 8.1 we prove that for any DAG G, $B\text{-}Peb(G)$ is equivalent to the total space of any I-RES-W$^-$ derivation of a certain variation of $Peb^1(G)$ formulas. Following this we show how to dispense with weakening and prove the equivalent result for I-RES. In Section 8.2 we put these equivalence results to use in order to prove the $\mathcal{PSPACE}$-Completeness of various forms of the I-RES Total Space Problem. Finally, in Section 9.2 we prove another corollary to our equivalence results, namely an optimal I-RES size / total space tradeoff.

8.1 Equivalence of Black Pebbling & Input Resolution Total Space

We shall prove that for any DAG G, $B\text{-}Peb(G)$ (with sliding) is almost exactly equivalent to the minimum total space of any I-RES-W$^-$ derivation from $Peb^1(G)^*$ with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$.

In order to proceed, we must first define the I-RES-W$^-$ proof system, which is simply I-RES with an added weakening rule, as well as the $Peb^1(G)^*$ formulas, which are a slight modification of the $Peb^1(G)$ formulas:

Definition 8.1 (I-RES-W$^-$). I-RES-W$^-$ *is identical to the* I-RES *proof system from Definition 2.3 except that it has the following additional rule:*

4. replacing one of the clauses $C \in \mathbb{C}_i$ with the clause $C \cup \{\neg x\}$, which was obtained by weakening C with an arbitrary negative literal $\neg x$.

The $Peb^1(G)^*$ formulas, closely related to the $Peb^1(G)$ formulas from Definition 2.7, are defined as follows:

Definition 8.2 ($Peb^1(G)^*$)**.** *In the case of binary DAGs without OR gates, in order to ensure that our resulting formula is a 3-CNF formula, we define $Peb^1(G)^*$ to be just like $Peb^1(G)$ except that we include dummy literals $\neg\alpha$ and $\neg\beta$ in each singleton clause so that each source clause $\{s\}$ becomes $\{s, \neg\alpha, \neg\beta\}$ and the target clause becomes $\{\neg t, \neg\alpha, \neg\beta\}$. Note that there are no positive instances of α or β, so a proof of $Peb^1(G) \vdash \emptyset$ will correspond exactly with a proof of $Peb^1(G)^* \vdash \{\neg\alpha, \neg\beta\}$.*

We make use of the $Peb^1(G)^*$ because unlike the $Peb^1(G)$ formulas, they are in 3-CNF.

We shall first show that the equivalence between pebbling number and I-RES space holds for I-RES-W$^-$, and then we will show that it also holds for standard I-RES by demonstrating how to dispense with weakening. In order to prove these results, we first need to define two very similar concepts. The first is a way of documenting the moves made during a pebbling game. We refer to such a record as a 'pebbling history':

Definition 8.3 (Pebbling Histories & Strategies). *A pebbling history on a monotone circuit G with j moves is a sequence of pairs $H = (B_0, W_0), (B_1, W_1), ..., (B_j, W_j)$ in which each (B_i, W_i) pair completely describes which nodes of G have pebbles on them at step i of the pebbling game; B_i is the set of nodes having black pebbles on them, and W_i is the set of nodes having white pebbles on them at time i. A step in the pebbling game is defined as being one move by the player consisting of the removal of a pebble, the placement of a pebble, or the sliding of a pebble (if sliding is allowed).*

When dealing with pure black or pure white pebbling histories, we allow $H = S_0, S_1, ..., S_j$ to be a sequence of sets in which each S_i is the set of nodes at time i having pebbles on them. This simplifies our notation, since pairs are not necessary in this case.

A 'pebbling strategy' is a history which has met the termination condition of the game.

The second concept which we need to define before proceeding is called an I-RES refutation 'Backbone', and it is very similar to the idea of a one-colour pebbling strategy:

Definition 8.4 (I-RES Backbone). *Let T be the tree underlying an I-RES refutation π of a formula F. The Backbone B of π is the linear portion of T starting at π's top clause C_0 and ending at the goal clause C_k, with all of the remaining clauses (which are all input clauses) removed from T. B can therefore be written as a sequence of clauses $B = B_0, B_1, ..., B_k$. Depending on our application we may consider B_0 to be the top clause and B_k to be the goal clause, or the other way around.*

8.1.1 The Equivalence of Black Pebbling & Input Total Space With Weakening

Our first lemma proves the forward direction of our equivalence and states that it is possible to take a pure black pebbling strategy of an arbitrary DAG G and from it build an I-RES-W$^-$ derivation from $Peb^1(G)^*$ with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ in which the backbone perfectly encodes the strategy.

The intuition behind this translation is illustrated by the following example: Consider the pyramid graph G shown below in Figure 2. We will use the labels on its nodes to correspond to the variable names in its pebbling formula, so $Peb^1(G)^* = (4 \vee \neg\alpha \vee \neg\beta) \wedge (5 \vee \neg\alpha \vee \neg\beta) \wedge (6 \vee \neg\alpha \vee \neg\beta) \wedge (\neg 4 \vee \neg 5 \vee 2) \wedge (\neg 5 \vee \neg 6 \vee 3) \wedge (\neg 2 \vee \neg 3 \vee 1) \wedge (\neg 1 \vee \neg\alpha \vee \neg\beta)$.

Figure 3 below shows how to translate the pure black pebbling strategy $S_0, S_1, ..., S_8$ for G into an I-RES-W$^-$ derivation of from $Peb^1(G)^*$ with top clause $\{\neg 1, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ in which the backbone perfectly encodes the strategy:

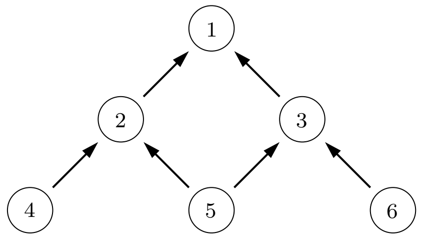

Figure 2: An Example of a Pyramid Graph; The target node is vertex 1.

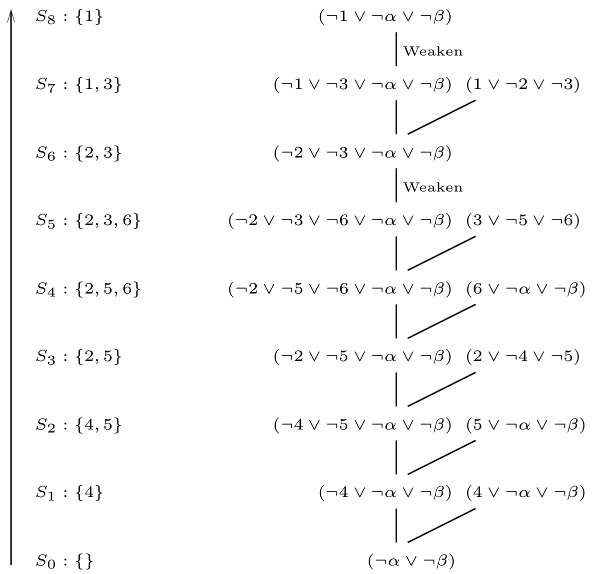

Figure 3: A Pure-Black Pebbling Strategy for G (Left) and its Corresponding I-RES-W$^-$ Refutation (Right)

More formally, translating a pebbling strategy into an I-RES-W$^-$ proof can be carried out according to the following lemma:

Lemma 8.5. *For any DAG G with target node t, if G has a pure black k-pebbling strategy (using sliding) $S = S_0, S_1, S_2..., S_{j-1}, S_j$ where $t \in S_j$, then $Peb^1(G)^*$ has an* I-RES-W$^-$ *derivation π with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ in which π has a backbone $B = B_0, B_1, ..., B_{j-1}, B_j$ (where B_j is the top clause and B_0 is the goal clause) such that for all $0 \leq i \leq j$, $B_i = \bigcup_{v \in S_i} \{\neg v\} \cup \{\neg\alpha, \neg\beta\}$.*

Proof: Let G be any arbitrary DAG with target node t and let $S = S_0, S_1, S_2..., S_{j-1}, S_j$ be any pure black k-pebbling strategy (using sliding) of G. Note that $S_0 = \emptyset$ and $t \in S_j$. We will show that there is a corresponding I-RES-W$^-$ derivation with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ by induction on the number of steps in the pebbling strategy S. It is useful to picture the pebbling strategy as a linear sequence of sets drawn with the first step at the bottom, and the last step at the top. We will construct the I-RES-W$^-$ proof and backbone from the bottom (goal clause) to the top (top clause). Note that because the dummy literals $\neg\alpha$ and $\neg\beta$ are present in every clause of the proof backbone, each backbone clause has a width that is two greater than the corresponding step in the pebbling strategy.

Basis: Step 0 of the pebbling strategy contains no pebbles. Therefore $S_0 = \emptyset$. The corresponding clause in our backbone is the clause that we are trying to derive. Therefore $B_0 = \{\neg\alpha, \neg\beta\}$. Since $\bigcup_{v \in S_0}\{\neg v\} = \emptyset$, it is clear that $B_0 = \bigcup_{v \in S_0}\{\neg v\} \cup \{\neg\alpha, \neg\beta\}$.

Induction Hypothesis: Suppose that we have been able to translate our pebbling strategy up to and including step i to I-RES proof steps in which the backbone clauses encode the pebbling strategy. More specifically, $B_i = \bigcup_{v \in S_i}\{\neg v\} \cup \{\neg\alpha, \neg\beta\}$, holds for step S_i.

Induction Step: We now show how to translate step $i+1$ of our pebbling strategy into a corresponding backbone clause. More specifically, we need to show that $B_{i+1} = \bigcup_{v \in S_{i+1}}\{\neg v\} \cup \{\neg\alpha, \neg\beta\}$ holds for S_{i+1}, step $i+1$ in our pebbling strategy. Pebbling step S_{i+1} could have come from step S_i in one of exactly three ways:

1. By pebbling a source node s.
2. By removing a pebble from vertex u.
3. If nodes a and b are pebbled predecessors of c, by sliding one of the pebbles from a or b to c.

Case 1: In this case, $S_{i+1} = S_i \cup \{s\}$. Let $B_{i+1} = B_i \cup \{\neg s\}$. Then we can derive B_i from B_{i+1} by resolving B_{i+1} with the input clause $\{s, \neg\alpha, \neg\beta\}$, so this is a valid resolution step resolving on the variable s. By our induction hypothesis, $B_i = \bigcup_{v \in S_i}\{\neg v\} \cup \{\neg\alpha, \neg\beta\}$, but $B_{i+1} = B_i \cup \{\neg s\}$, so $B_{i+1} = \bigcup_{v \in S_i}\{\neg v\} \cup \{\neg\alpha, \neg\beta\} \cup \{\neg s\}$. Since $S_{i+1} = S_i \cup \{s\}$, we know that $B_{i+1} = \bigcup_{v \in S_{i+1}}\{\neg v\} \cup \{\neg\alpha, \neg\beta\}$, as required.

Case 2: In this case, $S_{i+1} = S_i - \{u\}$. Let $B_{i+1} = B_i - \{\neg u\}$. Then we can derive B_i from B_{i+1} by weakening B_{i+1} to introduce $\{\neg u\}$, so this is a valid resolution step. By our induction hypothesis, $B_i = \bigcup_{v \in S_i}\{\neg v\} \cup \{\neg\alpha, \neg\beta\}$, but $B_{i+1} = B_i - \{\neg u\}$, so $B_{i+1} = \bigcup_{v \in S_i}\{\neg v\} \cup \{\neg\alpha, \neg\beta\} - \{\neg u\}$. Since $S_{i+1} = S_i - \{u\}$, we know that $B_{i+1} = \bigcup_{v \in S_{i+1}}\{\neg v\} \cup \{\neg\alpha, \neg\beta\}$, as required.

Case 3: Without loss of generality, assume that we are sliding the pebble from a to c. In this case, $S_{i+1} = S_i - \{a\} \cup \{c\}$. Let $B_{i+1} = B_i - \{\neg a\} \cup \{\neg c\}$. Then we can derive B_i from B_{i+1} by resolving B_{i+1} on variable a with the input clause $\{\neg a, \neg b, c\}$, so this is a valid I-RES-W$^-$ step resolving on the variable c. By our induction hypothesis, $B_i = \bigcup_{v \in S_i}\{\neg v\} \cup \{\neg\alpha, \neg\beta\}$, but $B_{i+1} = B_i - \{\neg a\} \cup \{\neg c\}$, so $B_{i+1} = \bigcup_{v \in S_i}\{\neg v\} \cup \{\neg\alpha, \neg\beta\} - \{\neg a\} \cup \{\neg c\}$. Since $S_{i+1} = S_i - \{a\} \cup \{c\}$, we know that $B_{i+1} = \bigcup_{v \in S_{i+1}}\{\neg v\} \cup \{\neg\alpha, \neg\beta\}$, as required.

Therefore, in all cases, there exists a valid next step in the backbone such that $B_{i+1} = \bigcup_{v \in S_{i+1}}\{\neg v\} \cup \{\neg\alpha, \neg\beta\}$, so by induction we can translate a pure black pebbling into an I-RES-W$^-$ proof in which the backbone perfectly encodes the pebbling strategy. In order to ensure that the I-RES-W$^-$ proof has top clause $\{\neg t, \neg\alpha, \neg\beta\}$, we may have to apply several weakenings in reverse. This is because the pebbling strategy may end with more than just the target node pebbled, in which case we would have to remove the literals corresponding to these superfluous pebbles in order to ensure that the top clause is indeed at the top of our I-RES-W$^-$ proof. □

Our next lemma shows the opposite direction, namely that it is possible to take any arbitrary I-RES-W$^-$ derivation π and translate it into a pebbling strategy which uses at most two fewer pebbles than π's backbone width.

Once again, we use the pyramid graph G from Figure 2 above as our example. Figure 4 below shows how to translate the I-RES-W$^-$ derivation of from $Peb^1(G)^*$ with top clause $\{\neg 1, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ into a pure white pebbling strategy $S_0, S_1, ..., S_{10}$ in which the strategy perfectly encodes backbone of the proof. In fact, weakening is never required in this example, so it actually translates an I-RES derivation to a pebbling strategy. Note how each Resolution step may require up to two corresponding pebbling moves.

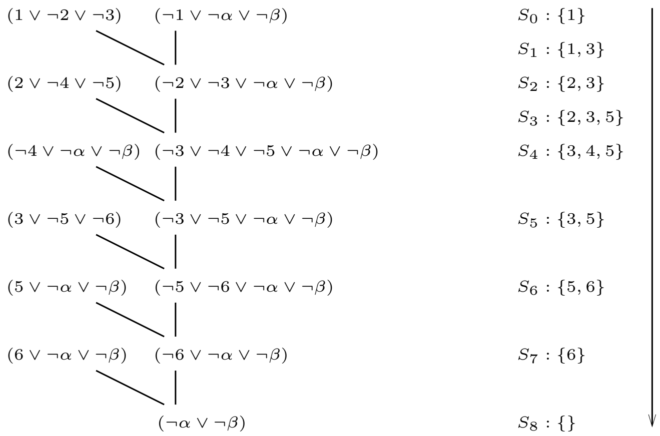

Figure 4: An I-RES-W$^-$ Derivation of $\{\neg\alpha, \neg\beta\}$ from $Peb^1(G)^*$ (Left) and its Corresponding Pure White Pebbling Strategy (Right)

More formally, translating a I-RES-W$^-$ proof into a pebbling strategy can be carried out according to the following lemma:

Lemma 8.6. *For any DAG G with target node t, if $Peb^1(G)^*$ has an I-RES-W$^-$ derivation π with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ in which π has a backbone $B = B_0, B_1, ..., B_j$ (where B_0 is the top clause and B_j is the goal clause), then there exists a pure white pebbling strategy (using sliding) $S = S_0, S_1, S_2..., S_l$ where l is $\leq 2j$ with $S_0 = \{t\}$ and $S_l = \emptyset$ such that it is possible to translate each B_i into either one pebbling step S_q such that $S_q = \bigcup_{\neg v \in B_i}\{v\} - \{\neg\alpha, \neg\beta\}$, or to translate B_i into two consecutive pebbling steps S_{q_1}, S_{q_2} such that $|S_{q_1}| \leq |B_i| - 2$ and $S_{q_2} = \bigcup_{\neg v \in B_i}\{v\} - \{\neg\alpha, \neg\beta\}$.*

Proof: Let G be any arbitrary DAG with target node t and let π be an I-RES-W$^-$ derivation with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ in which π has a backbone of width $k + 2$. We will show how to take this backbone and use it to create a pure white pebbling strategy which uses at most k pebbles. It is useful to picture π with the top clause at the top and the goal clause at the bottom. We shall move down the backbone of π and show how to build a corresponding pure white pebbling strategy. Note that every clause in the backbone contains the dummy literals $\neg\alpha$ and $\neg\beta$ and that this accounts for the '+ 2' in $k + 2$.

Basis: Consider the top clause of π, $B_0 = \{\neg t, \neg\alpha, \neg\beta\}$; we create the equivalent first step of a pebbling by placing a white pebble on node t to create pebbling step $S_0 = \{t\}$. Therefore $S_0 = \bigcup_{\neg v \in B_0}\{v\} - \{\neg\alpha, \neg\beta\}$, as required.

Induction Hypothesis: Suppose that we have been able to translate our backbone into a pebbling strategy up to and including backbone clause B_i and that B_i was translated to S_p such that $S_p = \bigcup_{\neg v \in B_i}\{v\} - \{\neg\alpha, \neg\beta\}$.

Induction Step: We will now show how to translate backbone clause B_{i+1} to create either one or two corresponding pebbling steps. Backbone clause B_{i+1} could have come from B_i in one of exactly three ways:

1. By resolving B_i with a source clause $\{u, \neg\alpha, \neg\beta\}$ (resolving on variable u).
2. By resolving B_i with a propagation clause $\{\neg a, \neg b, c\}$ (resolving on variable c).
3. By weakening B_i to introduce a negative literal $\neg u$.

Case 1: By our induction hypothesis, we have translated backbone clause B_i into pebbling step S_p such that $S_p = \bigcup_{\neg v \in B_i} \{v\} - \{\neg\alpha, \neg\beta\}$. We now resolve B_i with the input clause $\{u, \neg\alpha, \neg\beta\}$ on variable u to create $B_{i+1} = B_i - \{\neg u\}$. Our corresponding pebbling move is to take S_p and create S_q by removing the white pebble on node u. Therefore $S_q = \bigcup_{\neg v \in B_i} \{v\} - \{\neg\alpha, \neg\beta\} - \{\neg u\}$, but $B_{i+1} = B_i - \{\neg u\}$, so $S_q = \bigcup_{\neg v \in B_{i+1}} \{v\} - \{\neg\alpha, \neg\beta\}$, as required.

Case 2: By our induction hypothesis, we have translated backbone clause B_i into pebbling step S_p such that $S_p = \bigcup_{\neg v \in B_i} \{v\} - \{\neg\alpha, \neg\beta\}$. We now resolve B_i with a propagation clause $\{\neg a, \neg b, c\}$ on variable c to create $B_{i+1} = B_i \cup \{\neg a, \neg b\} - \{\neg c\}$. We make two corresponding pebbling moves. The first is to place a white pebble on node a. Therefore $S_{p_1} = S_p \cup \{a\}$, so the size of our pebbling state has increased by 1. Since $B_{i+1} = B_i \cup \{\neg a, \neg b\} - \{\neg c\}$, and neither a nor b were pebbled in S_p (which has corresponding backbone clause $S_p = \bigcup_{\neg v \in B_i} \{v\} - \{\neg\alpha, \neg\beta\}$ of size $|B_i| - 2$), this means that the variables a and b do not occur in B_i, so $|B_{i+1}| = |B_i| + 2$. Therefore $|S_{p_1}| \leq |B_{i+1}| - 2$.

The second and final pebbling move is to slide the white pebble on c to b. Therefore $S_{p_2} = S_p \cup \{a, b\} - \{c\}$. But $S_p = \bigcup_{\neg v \in B_i} \{v\} - \{\neg\alpha, \neg\beta\}$, so $S_{p_2} = \bigcup_{\neg v \in B_i} \{v\} - \{\neg\alpha, \neg\beta\} \cup \{a, b\} - \{c\}$, but $B_{i+1} = B_i \cup \{\neg a, \neg b\} - \{\neg c\}$, so $S_{p_2} = \bigcup_{\neg v \in B_{i+1}} \{v\} - \{\neg\alpha, \neg\beta\}$, as required.

Case 3: By our induction hypothesis, we have translated backbone clause B_i into pebbling step S_p such that $S_p = \bigcup_{\neg v \in B_i} \{v\} - \{\neg\alpha, \neg\beta\}$. We now weaken B_i to introduce the negative literal $\neg u$, so $B_{i+1} = B_i \cup \{\neg u\}$. Therefore our corresponding pebbling move is to take S_p and create S_q by placing a white pebble on u. Therefore $S_q = S_p \cup \{u\} = \bigcup_{\neg v \in B_i} \{v\} - \{\neg\alpha, \neg\beta\} \cup \{u\} = \bigcup_{\neg v \in B_{i+1}} \{v\} - \{\neg\alpha, \neg\beta\}$, as required.

Therefore, in all three cases, there exists a valid next step in the pebbling such that $S_q = \bigcup_{\neg v \in B_{i+1}} \{v\} - \{\neg\alpha, \neg\beta\}$, so by induction we can translate an $\mathsf{I\text{-}RES\text{-}W}^-$ backbone B into a pure white pebbling strategy S such that S perfectly encodes B. □

The following corollary combines the previous two Lemmas in order to prove an equivalence between the black pebbling number of DAG G and the $\mathsf{I\text{-}RES\text{-}W}^-$ total space of $Peb^1(G)^*$.

Corollary 8.7. *For any DAG G with target node t, G has a pure black k-pebbling strategy (using sliding) $S = S_0, S_1, S_2 ..., S_j$ with j steps where $t \in S_j$ and j is $O(l)$ if and only if $Peb^1(G)^*$ has an $\mathsf{I\text{-}RES\text{-}W}^-$ derivation π with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ in which π has total space $k + 5$ and the backbone of π contains l clauses where l is $O(j)$.*

Proof: Let G be any arbitrary DAG with target node t.

$\Rightarrow$ Suppose that G has a pure black k-pebbling strategy (using sliding) $S = S_0, S_1, S_2 ..., S_j$ with j steps where $t \in S_j$ and j is $O(q)$. By Lemma 8.5, it is possible to translate this pebbling strategy into an $\mathsf{I\text{-}RES\text{-}W}^-$ derivation π of $Peb^1(G)^*$ with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ such that the backbone of π contains l clauses, each of which has width at most $k + 2$. However, since $Peb^1(G)^*$ is a 3-CNF formula, each of the input clauses used to resolve with clauses in the backbone has width 3. Since input refutations only require there to be two clauses in memory at any time, π therefore has total space $k + 5$ and its backbone contains $O(j)$ clauses.

$\Leftarrow$ Suppose that $Peb^1(G)^*$ has an $\mathsf{I\text{-}RES\text{-}W}^-$ derivation π of $\{\neg\alpha, \neg\beta\}$ with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ in which π has total space $k + 5$ and π's backbone contains l clauses where l is $O(j)$. Since $Peb^1(G)^*$ is a 3-CNF formula, π has a backbone with width $k+2$. By Lemma 8.6, it is possible to translate this backbone into a pure white pebbling strategy containing l steps which uses at most k pebbles. But by Lemma 7.2, it is possible to convert this pure white pebbling strategy into a pure black pebbling strategy with $O(l)$ steps which uses exactly the same number of pebbles. Therefore G has a pure black k-pebbling strategy (using sliding) with $O(l)$ steps. □

8.1.2 The Equivalence of Black Pebbling & Input Derivation Total Space

We now show that the weakenings in any I-RES-W$^-$ proof π can be removed to turn it into a weakening-free proof π' with the same backbone width.

Lemma 8.8. *For any unsatisfiable set of Horn clauses F, any $C \in F$, and any goal clause D, there exists an* I-RES-W$^-$ *proof π from F with top clause C and goal clause D such that the backbone width of π is at most k and the size of π is s if and only if there exists an* I-RES *proof π' from F with top clause C and goal clause $D^{'} \subseteq D$ such that the backbone width of π' is at most k and the size of π' is $\leq s$.*

Proof: Let F be any unsatisfiable set of Horn clauses, let C be any arbitrary initial clause in F, and let D be any arbitrary goal clause.

$\Rightarrow$ Suppose that there exists an I-RES-W$^-$ proof π from F with top clause C and goal clause D such that the backbone width of $\pi \leq k$ and the size of π is $O(s)$. In order to show that there exists an I-RES proof π' from F with top clause C and goal clause $D^{'} \subseteq D$ such that the backbone width of $\pi' \leq k$ and the size of π' is $O(s)$, we will show how to translate π into π' without losing any of these properties. We shall first translate π into π^* by removing all of the weakenings in π as well as all applications of the Resolution rule which no longer apply due to the fact that weakened variables (and all other variables derived by Resolving on them) are now missing. It is important to note that π^* is not necessarily a legitimate I-RES proof, because our translation method may cause two successive clauses in the backbone of π to be translated into successive duplicate clauses in the backbone of π^*.

We translate π into π^* via induction on the depth (the top clause having the smallest depth) of π according to the following inductive procedure / proof. We shall translate each clause C_i in the backbone of π into a clause C_i^* in the backbone of π^* such that $\forall i\ C_i \subseteq C_i^*$.

Basis: The top clause in π is C. Set the top clause in π^* to be C as well. Therefore $C_0 \subseteq C_0^*$, as required.

Induction Hypothesis: Suppose that $C_i \subseteq C_i^*$.

Induction Step: We now show how to translate π into π^* such that $C_{i+1} \subseteq C_{i+1}^*$:

Case 1: Suppose that C_{i+1} came from C_i in π via weakening. In this case, simply omit this weakening step in π^*. Since $C_i \subseteq C_i^*$ by our induction hypothesis, and C_{i+1} has grown, whereas C_{i+1}^* has not, we can conclude that $C_{i+1} \subseteq C_{i+1}^*$.

Case 2a: Suppose that C_{i+1} came from C_i in π by resolving C_i with the input clause I on variable x, but this resolution step is not possible in π^* because $x \notin C_i^*$. In this case, simply omit this resolution step in π^*. We know by the induction hypothesis that $C_i \subseteq C_i^*$, and since $x \notin C_i^*$, we can conclude that $C_{i+1} \subseteq C_{i+1}^*$.

Case 2b: Suppose that C_{i+1} came from C_i in π by resolving C_i with the input clause I on variable x, and $x \in C_i^*$. In this case, simply perform the same resolution step in π^*; i.e. resolve C_i^* with I to get C_{i+1}^*. By the induction hypothesis that $C_i \subseteq C_i^*$, so $C_{i+1} \subseteq C_{i+1}^*$.

Therefore, in all cases, $C_{i+1} \subseteq C^*_{i+1}$, showing that by induction, $D \subseteq D^*$, where D is the goal clause of π and D^* is the goal clause of π^*. Finally, since π^* might contain duplicate clauses along its backbone, we simply remove all but one out of each group of duplicates to create π', which is now a legitimate I-RES proof. Therefore there exists an I-RES proof π' from F with top clause C and goal clause $D^{'} \subseteq D$ such that the backbone width of $\pi' \leq k$ and the size of π' is $\leq s$, as required.

$\Leftarrow$ Suppose there exists an I-RES proof π' from F with top clause C and goal clause $D^{'} \subseteq D$ such that the backbone width of $\pi' \leq k$ and the size of π' is $\leq s$. Since every I-RES proof is an I-RES-W$^-$ proof, π' is also an I-RES-W$^-$ proof with goal clause $D^{'} \subseteq D$. We convert π' to π by weakening $D^{'}$ to get D. Therefore there exists an I-RES-W$^-$ proof π from F with top clause C and goal clause D such that the backbone width of $\pi \leq k$ and the size of π is $O(s)$, as required. □

The following corollary is identical to Corollary 8.7 except that instead of proving an equivalence between the black pebbling number of a DAG G and the I-RES-W$^-$ total space of $Peb^1(G)^*$, here we use our weakening removal Lemma from above to show that this equivalence also holds for I-RES without weakening:

Corollary 8.9. *For any DAG G with target node t, G has a pure black k-pebbling strategy (using sliding) $S = S_0, S_1, S_2 ..., S_j$ with j steps where $t \in S_j$ and j is $O(l)$ if and only if $Peb^1(G)^*$ has an* I-RES *derivation π with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ in which π' has total space $k+5$ and size l where l is $O(j)$.*

Proof: Let G be any arbitrary DAG with target node t.

$\Rightarrow$ Suppose that G has a pure black k-pebbling strategy (using sliding) $S = S_0, S_1, S_2 ..., S_j$ with j steps where $t \in S_j$ and j is $O(l)$. By Corollary 8.7, $Peb^1(G)^*$ has an I-RES-W$^-$ derivation π with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ in which π has total space $k+5$ and size l where l is $O(j)$. Because $Peb^1(G)^*$ is a 3-CNF formula, π's backbone has width $\leq k+2$. By Lemma 8.8 there exists an I-RES proof π' from F with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $D^{'} \subseteq \{\neg\alpha, \neg\beta\}$ such that the backbone width of $\pi' \leq k+2$ and the size of π' is $O(j)$. Since F does not include any positive instances of the variables α or β, we can conclude that $D^{'} = \{\neg\alpha, \neg\beta\}$. Therefore, since $Peb^1(G)^*$ is a 3-CNF formula, π' has total space $k+5$ and size $O(j)$, with the same top and goal clauses as π.

$\Leftarrow$ Suppose that $Peb^1(G)^*$ has an I-RES derivation π' with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ in which π' has total space $k+5$ and size l where l is $O(j)$. But every I-RES proof is an I-RES-W$^-$ proof, so there exists an I-RES-W$^-$ derivation π with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ in which π has total space $k+5$ and size l, which is $O(l)$. Therefore, by Corollary 8.7 G has a pure black k-pebbling strategy. □

8.2 The $\mathcal{PSPACE}$-Completeness of Input Derivation Total Space

This section contains some immediate corollaries to the previous results. Here we prove the $\mathcal{PSPACE}$-Completeness of three slightly different versions of the I-RES derivation total space problem, which is surprising, since I-RES seems like such a trivial Resolution refinement.

The I-RES derivation total space problem (ITS) is defined as follows: The input instance is (F, C, D, k), where F is an unsatisfiable set of clauses, C is any clause in F, D is a goal clause, and k is an integer. The problem is to determine if there exists an I-RES derivation from F with top clause C, goal clause D, and total space at most k.

We shall prove the $\mathcal{PSPACE}$-Completeness of three different versions of this problem:

1. The set F is restricted to containing only Horn clauses, and instead of asking whether there exists an I-RES derivation from F with top clause C, goal clause D, and total space at most k, we ask whether there exists an I-RES-W$^-$ derivation with negative weakening from F with top clause C, goal clause D, and total space at most k. This is the input derivation with negative weakening total space problem, and we shall refer to the language associated with it as $HWITS$, where the 'HW' stands for '**H**orn with Negative **W**eakening'.

2. This version of the problem is almost identical to the previous one; the set F is still restricted to containing only Horn clauses, but this time we are dealing with I-RES rather than I-RES-W$^-$. We shall refer to the language associated with this version of the problem as $HITS$.

3. Again, this version of the problem is almost identical to the previous one, only this time there is no restriction on F. We shall refer to the language associated with this version of the problem as ITS.

8.2.1 $\mathcal{PSPACE}$-Completeness of the Input Total Space Problem with Weakening

The language $HWITS$ is formally defined as follows:

Definition 8.10 ($HWITS$)**.** $HWITS = \{(F, C, D, k) \mid$ *F is a Horn formula for which there exists an* I-RES-W$^-$ *refutation with top clause C, goal clause D, and total space at most k*$\}$

The $\mathcal{PSPACE}$-Completeness of $HWITS$ follows from the equivalence between the black pebbling number of DAG G and the I-RES-W$^-$ total space of $Peb_1(G)^*$ given in Corollary 8.7 from the previous section. This language is formally defined as follows:

Theorem 8.11. *$HWITS$ is $\mathcal{PSPACE}$-Complete under logspace reducibility.*

Proof: We first show that $HWITS \in \mathcal{PSPACE}$. We are given an input instance (F, $C \in F$, D, k) and asked to determine if F has an I-RES-W$^-$ proof π with top clause C and goal clause D such that π's total space is bounded above by k. Since we are dealing with input refutations, we know that if such a π exists, then each of its configurations contains no more than two clauses. Since each clause contains at most n literals, it is clear that every configuration in π takes only polynomial space.

Our algorithm proceeds as follows: Start with a configuration $\mathbb{C}_0 = \{C\}$. Guess configuration $\mathbb{C}_1$, check to ensure that it follows from $\mathbb{C}_0$ by a legal I-RES-W$^-$ step, and erase configuration $\mathbb{C}_0$. Next, guess configuration $\mathbb{C}_2$, check to make sure that it follows from $\mathbb{C}_1$, and erase configuration $\mathbb{C}_1$. Continue this way until the goal clause has been derived. Note that at any time, there are only two configurations in memory. But since we are dealing with input refutations, we know that each configuration contains no more than two clauses. Since each clause contains at most n literals, it is clear that our non-deterministic algorithm requires polynomial space. This shows that $HWITS \in \mathcal{NPSPACE}$. Finally, we appeal to Savitch's Theorem [Sav70] to show that $HWITS \in \mathcal{PSPACE}$.

Next we show that $HWITS$ is $\mathcal{PSPACE}$-Hard by giving a reduction from the black pebbling number problem with sliding ($BLACK$-PEB) from [GLT80] which was shown to be $\mathcal{PSPACE}$-Complete. We are given an instance (G, k) where t is the target node in G and we wish to convert it to an instance (F, $C \in F$, D, k) such that $(G, k) \in BLACK$-PEB if and only if (F, $C \in F$, D, k) $\in HWITS$. Our reduction proceeds as follows: Take (G, k) and output ($Peb^1(G)^*$, $\{\neg t, \neg\alpha, \neg\beta\}$, $\{\neg\alpha, \neg\beta\}$, $k + 5$), which is clearly a logspace reduction. The proof of correctness for this reduction is given by Corollary 8.7. $\square$

8.2.2 The $\mathcal{PSPACE}$-Completeness of the Horn Input Total Space Problem

The language $HITS$ is formally defined as follows:

Definition 8.12 ($HITS$)**.** $HITS = \{(F, C, D, k) \mid$ *F is a Horn formula for which there exists an* I-RES *refutation with top clause C, goal clause D, and total space at most k*$\}$

The $\mathcal{PSPACE}$-Completeness of $HITS$ follows immediately from the equivalence between the black pebbling number of DAG G and the I-RES total space of $Peb_1(G)^*$ given in Corollary 8.9 from the previous section.

Corollary 8.13. *$HITS$ is $\mathcal{PSPACE}$-Complete under logspace reducibility.*

Proof: To show that $HITS \in \mathcal{PSPACE}$, we simply use the same algorithm given in Theorem 8.11.

To show that $HITS$ is $\mathcal{PSPACE}$-Hard, we give a reduction from $BLACK\text{-}PEB$. As in Theorem 8.11, take (G, k) and output $(Peb_1(G)^*,\ \{\neg t, \neg\alpha, \neg\beta\},\ \{\neg\alpha, \neg\beta\},\ k+5)$, which is clearly a logspace reduction, and its proof of correctness is given by Corollary 8.9. □

8.2.3 The $\mathcal{PSPACE}$-Completeness of the Input Total Space Problem

The language ITS is formally defined as follows:

Definition 8.14 (ITS). $ITS = \{(F, C, D, k) \mid F$ *is a formula for which there exists an* I-RES *refutation with top clause C, goal clause D, and total space at most k*$\}$

The $\mathcal{PSPACE}$-Completeness of ITS follows trivially from the $\mathcal{PSPACE}$-Completeness of ITS:

Corollary 8.15. *ITS is $\mathcal{PSPACE}$-Complete under logspace reducibility.*

Proof: To show that $ITS \in \mathcal{PSPACE}$, we simply use the same algorithm given in Theorem 8.11.

To show that ITS is $\mathcal{PSPACE}$-Hard, we reduce from $HITS$. Take the input F for $HITS$ and check to see if it is a set of Horn clauses. If not, then output a pre-chosen formula which is not in ITS. Otherwise, simply output F. Clearly this is a logspace reduction, and its correctness is immediate since Horn formulas are just a special case of what an ITS solver can decide. □

9 Related Complexity Results

The $\mathcal{PSPACE}$-Completeness of I-RES total space has some interesting corollaries:

9.1 The $\mathcal{PSPACE}$-Completeness of the Input Derivation Width Problem

One interesting point of note is that I-RES total space and width are virtually identical. More specifically, for every k-CNF formula, $w(F \vdash_{\text{I-RES}} D) = TS(F \vdash_{\text{I-RES}} D) - k$.

The $Peb_1(G)^*$ formulas are 3-CNF formulas, so by Corollary 8.9, the width of any I-RES derivation of goal clause $\{\neg\alpha, \neg\beta\}$ from $Peb_1(G)^*$ with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ is $k+2$. This implies that the I-RES derivation width problem is therefore also $\mathcal{PSPACE}$-Complete:

Corollary 9.1. *Given a formula F, a top clause C, goal clause D, and integer k, the problem of determining if there exists an* I-RES *derivation of D from F with top clause C with width at most k is $\mathcal{PSPACE}$-Complete under logspace reducibility.*

9.2 Optimal Size / Total Space Tradeoffs For Input Resolution

In this section we describe another corollary to the results in Section 8.1. This result also relies closely on one of the most surprising facts concerning pebbling, namely that there exist infinite families of DAGs which take an exponential amount of time to be pebbled with the minimum number of pebbles, but if one is willing to use just one or two more pebbles, then they can be pebbled in linear time. The earliest known example of this phenomenon can be found in [Lin78], where the author gives an infinite family of monotone circuits such that pebbling any of them with the minimum number of pebbles requires $2^{\Omega(n^{1/3})}$ time, where n is the number of nodes in the circuit. However, if given only two more pebbles, the amount of time required drops exponentially to only $O(n)$, giving a massive pebble / pebbling time tradeoff.

This result was later improved by Gilbert, Lengauer, and Tarjan:

Lemma 9.2 ([GLT80]). *There exists an infinite family of DAGs $\mathcal{G}$, such that pebbling any $G \in \mathcal{G}$ with the minimum number of pebbles takes $\Omega(2^n)$ time, but if only one more pebble is used, then the amount of time required to pebble G drops exponentially to only $O(n)$, where n is the number of vertices in G.*

Such an exponential separation at the cost of only one pebble is extraordinary, but since Corollary 8.9 gives an exact relationship between black pebbling and total space for I-RES derivations, it is possible to translate this amazing result from the world of pebbling over to Resolution, thereby giving a massive size / total space tradeoff for I-RES:

Corollary 9.3. *There exists an infinite family of formulas $\mathcal{F} = \{Peb^1(G)^* | G \in \mathcal{G}\}$ such that for every $F \in \mathcal{F}$, every I-RES derivation π of F with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ where π has the minimum required total space, has size $\Omega(2^n)$, where n is the number of variables in F. However, if only one more unit of total space is permitted, then the size of π drops to only $O(n)$.*

Proof: Let $\mathcal{G}$ be the family of DAGs from Lemma 9.2, and let F be any arbitrary formula from $\mathcal{F}$. Therefore $F = Peb^1(G)^*$ for some $G \in \mathcal{G}$. G can be pebbled with k but not with fewer than k pebbles. By Corollary 8.9, F has an I-RES derivation π with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ such that π requires total space $k + 5$, but does not have an I-RES derivation with total space less than $k + 5$. We know that to pebble G with k pebbles requires $\Omega(2^n)$ time, so by Corollary 8.9, π has a size of at least $\Omega(2^n)$. Similarly, we know that pebbling G with $k + 1$ pebbles requires $O(n)$ time, so again by Corollary 8.9, there exists a proof π' with total space $k + 5 + 1 = k + 6$ and size $O(n)$, as required. $\square$

Therefore, using just one extra unit of total space yields an exponential decrease in size, giving an optimal size / total space tradeoff. As with our $\mathcal{PSPACE}$-Completeness results from the previous section, this is more evidence that the computational complexity of I-RES is underestimated.

Acknowledgements

We would like to thank Toni Pitassi for her very helpful comments regarding this paper.

References

[ABMP01] M. Alekhnovich, S. Buss, S. Moran, and T. Pitassi. Minimum Propositional Proof Length is NP-Hard to Linearly Approximate. *Journal of Symbolic Logic*, 66:171 – 191, 2001.

[AD03] A. Atserias and V. Dalmau. A Combinatorial Characterization of Resolution Width. *Proceedings of the* 18^{th} *IEEE Conference on Computational Complexity*, 2003.

[AL86] R. Aharoni and N. Linial. Minimal Non-Two-Colorable Hypergraphs and Minimal Unsatisfiable Formulas. *Journal of Combinatorial Theory, Series A*, 43:196– 204, 1986.

[AR02] M. Alekhnovich and A. Razborov. Resolution is Not Automatizable Unless $\mathcal{W}[\mathcal{P}]$ is Tractable. *Proceedings of the* 42^{nd} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.

[BOM07] J. Buresh-Oppenheim and D. Mitchell. Minimum 2CNF Resolution Refutations in Polynomial Time. *Proceedings of the* 10^{th} *International Conference on Theory and Applications of Satisfiability Testing (SAT 2007)*, pages 300 – 313, 2007.

[BS02] E. Ben-Sasson. Size Space Tradeoffs For Resolution. *Proceedings of the* 34^{th} *ACM Symposium on the Theory of Computing (STOC)*, pages 457 – 464, 2002.

[BSIW04] E. Ben-Sasson, R. Impagliazzo, and A. Wigderson. Near Optimal Separation of Tree-like and General Resolution. *Combinatorica*, Vol. 24, No. 4:585 – 604, 2004.

[BSW01] E. Ben-Sasson and A. Wigderson. Short Proofs are Narrow -Resolution Made simple. *Journal of the Association for Computing Machinery*, 48:149–169, 2001. Preliminary Version: Proceedings of the 31^{st} Annual ACM Symposium on Theory of Computing (STOC), 1999, pages 517-526.

[Cha70] C. L. Chang. The Unit Proof and the Input Proof in Theorem Proving. *Journal of the Association for Computing Machinery*, Vol. 17, No. 4:698 – 707, 1970.

[CK01] P. Clote and E. Kranakis. *Boolean Functions and Computation Models*. Springer-Verlag, Berlin, 2001.

[Coo71] S.A. Cook. The Complexity of Theorem-Proving Procedures. *Proceedings of the* 3^{rd} *Annual ACM Symposium on the Theory of Computation*, pages 151 – 158, 1971.

[Coo73] S. A. Cook. An Observation on Time-Storage Tradeoff. *Proceedings of the* 5^{th} *Annual ACM Symposium on Theory of Computing (STOC)*, 1973.

[CS76] S. Cook and R. Sethi. Storage Requirements for Deterministic Polynomial Time Recognizable Languages. *Journal of Computer & System Sciences*, pages 25 – 37, 1976.

[DDB98] G. Davydov, I. Davydova, and H. Kleine Büning. An Efficient Algorithm for the Minimal Unsatisfiability Problem for a Subclass of CNF. *Annals of Mathematics and Artificial Intelligence*, 23:229 – 245, 1998.

[ET01] J. Esteban and J. Torán. Space Bounds for Resolution. *Information and Computation*, 171:84 – 97, 2001. Preliminary Version: Proceedings of the 16^{th} Annual Symposium on Theoretical Aspects of Computer Science (STACS), 1999, pages 551-561.

[ET03] J. Esteban and J. Torán. A Combinatorial Characterization of Treelike Resolution Space. *Information Processing Letters*, 87:295–300, 2003. Preliminary Version: Electronic Colloquium on Computational Complexity (ECCC) Report TR03-044, 2003.

[GLT80] J. R. Gilbert, T. Lengauer, and R. E. Tarjan. The Pebbling Problem is Complete in Polynomial Space. *SIAM Journal of Computing*, Vol. 9, No. 3:513 – 524, 1980.

[Hei81] F. Meyer Auf Der Heide. A Comparison of Two Variations of a Pebble Game on Graphs. *Theoretical Computer Science*, 13:315 – 322, 1981.

[HP07] P. Hertel and T. Pitassi. Exponential Time / Space Speedups for Resolution and the PSPACE-Completeness of Black-White Pebbling. *Proceedings of the* 48^{th} *Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, 2007.

[HU07] A. Hertel and A. Urquhart. Game Characterizations and the PSPACE-Completeness of Tree Resolution Space. In *Proceedings of the* 16^{th} *EACSL Annual Conference on Computer Science and Logic (CSL 2007)*, pages 527 – 541. Springer, 2007. Lecture Notes in Computer Science 4646.

[JL77] N.D. Jones and W.T. Laaser. Complete Problems For Deterministic Polynomial Time. *Theoretical Computer Science*, 3:105 – 117, 1977.

[Lin78] A. Lingas. A PSPACE-Complete Problem Related to a Pebble Game. In *Proceedings of the* 5^{th} *Colloquium on Automata, Languages and Programming*, pages 300 – 321, London, UK, 1978. Springer-Verlag.

[Nor06] J. Nordström. Narrow Proofs May Be Spacious: Separating Space and Width in Resolution. *Proceedings of the* 38^{th} *ACM Symposium on the Theory of Computing (STOC)*, pages 507–516, 2006. Preliminary Version: Electronic Colloquium on Computational Complexity (ECCC) Report TR05-066, 2005.

[Pap94] C. H. Papadimitriou. *Computational Complexity.* Addison Wesley Longman, New York, 1994.

[Pip80] N. Pippenger. Pebbling. *Proceedings of the* 5^{th} *IBM Symposium on Mathematical Foundations of Computer Science, Japan (Technical Report RC8528, IBM Watson Research Center)*, 1980.

[Pla84] D.A. Plaisted. Complete Problems in the First-Order Predicate Calculus. *Journal of Computer and System Sciences*, Vol. 29, No. 1:8 – 35, 1984.

[Sav70] W. Savitch. Relationships Between Nondeterministic and Deterministic Tape Complexities. *Journal of Computer and System Sciences*, 4:177 – 192, 1970.

[Sub04] K. Subramani. Optimal Length Tree-Like Resolution Refutations for 2SAT Formulas. *ACM Transactions on Computational Logic*, Vol. 5, No. 2:316 – 320, 2004.

[Urq06] A. Urquhart. Width Versus Size in Resolution Proofs. *Proceedings of the* 3^{rd} *Annual Conference on Theory and Applications of Models of Computation*, pages 79 – 88, 2006.