Journal on Satisfiability, Boolean Modeling and Computation 6 (2009) 141-164
IOS Press

Algorithms and Complexity Results for Input and Unit Resolution

Alexander Hertel [*] ahertel@cs.toronto.edu
Alasdair Urquhart urquhart@cs.toronto.edu
Department of Computer Science
University of Toronto
Canada

Abstract

In this paper we explore the complexity of various problems pertaining to Input Resolution. In the first part of this paper we survey a number of earlier results for Input Resolution, showing the tractability of various aspects of this proof system. In the second part, we prove the PSPACE-Completeness of both Input Resolution total space and width, as well as a massive size/total space tradeoff for Input Resolution. These results suggest that although Input Resolution is completely tractable with respect to certain complexity measures such as refutation size, when quantities such as space and width are considered, the system shows a surprising level of difficulty.

Keywords: *satisfiability, complexity, resolution, input resolution*

Submitted November 2007; revised June 2008; published May 2009

1. Introduction and Motivation

The Resolution (RES) proof system has been studied extensively, and is understood quite well. Because it forms the basis of most 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 too complicated even in RES, so 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 have both been very successful when used as the basis of SAT-solving algorithms. Of course, in some cases a RES refutation of minimal size may be of length exponential in the size of the input clauses, as follows from the basic results on the size of RES proofs, beginning with the result of Armin Haken [16]; the reader is referred to the textbook by Clote and Kranakis [9] for background in this area. However, in spite of these complexity results, we can nevertheless take as a

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

goal of automated theorem proving the development of algorithms capable of finding proofs that are only polynomially larger than the optimal. Proof systems in which this is possible are said to be *automatizable*.

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 I-RES proof system is more interesting than it may first appear. It is important both theoretically as well as practically because of a theorem by Chang [8] showing that a formula has an I-RES refutation if and only if it has a Unit Resolution (U-RES) refutation. This in part motivates the study of I-RES, since U-RES is an important subroutine found in many areas of computer science, not least of all in automated theorem proving and SAT-solving.

In this paper we prove a number of complexity results for I-RES including $\mathcal{P}$-Completeness, $\mathcal{NP}$-Completeness, $\mathcal{PSPACE}$-Completeness, and exponential tradeoff 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. Preliminary versions of the results in this paper can be found in [18].

Specialized definitions and concepts that 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 the textbooks by Clote and Kranakis [9] and Papadimitriou [22] 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 that 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 a formula 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.

Next, in Section 7 we prove an equivalence between I-RES total space and pebbling, followed immediately by its implications for complexity theory. We show that for any binary DAG G, its pebbling number is off by exactly a constant from the total space required by any I-RES proof of a slight modification to the formula $Peb(G)$. This equivalence allows us to prove our main results, namely the $\mathcal{PSPACE}$-Completeness of the I-RES total space problem, by reducing from the pebbling game on DAGs, itself shown to be $\mathcal{PSPACE}$-Complete by Gilbert, Lengauer, and Tarjan [15]. 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 8 we prove two interesting corollaries to the equivalence results in the previous section. 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 $O(n)$, once again contrasting strongly with the apparent simplicity of I-RES as suggested by the earlier tractability and automatizability results. Apart from being a massive size/space tradeoff, this is also a tradeoff similar to those explored by Ben-Sasson in [6] because it shows that for certain formulas, it is not possible to optimize both I-RES size and space at the same time.

2. Definitions

2.1 Resolution Proof, Size, and Width

A *literal* is a propositional variable x or its negation $\neg x$; we assume the identification $\neg\neg x = x$. A *clause* is a set of literals. 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)**.** *If $C \cup \{x\}$ and $D \cup \{\neg x\}$ are clauses, then the resolution rule allows us to derive the clause $C \cup D$ 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).*

In a RES proof (considered as a sequence of clauses), the first two clauses resolved on are called the *top clauses* of π. The final result of the proof, C_k, is called the *goal clause.*

Definition 2.2 (Resolution Size and 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, $w(\pi)$, is equal to the width of its widest clause. 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 that depends on the notion of configuration. This new definition of a RES proof was introduced independently by Esteban and Torán [13] and Alekhnovich, Ben-Sasson, Razborov and Wigderson [2]; Esteban and Torán's definition is a modification of an earlier definition due to Kleine Büning and Lettmann [7]. For more information on the history of proof complexity space research as well as several space results related to this paper, please refer to [18].

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. *Delete one or more of the clauses in $\mathbb{C}_i$;*
2. *Add the resolvent of two clauses of $\mathbb{C}_i$;*
3. *Add 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. Replace one of the clauses $C \in \mathbb{C}_i$ with the clause $C \cup D$, where D is a set of negative literals.

*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 (*U-RES*) proof, and if $\emptyset \in \mathbb{C}_k$, then π is a refutation.*

The next definition is intended to capture the idea that a RES proof might be very long (perhaps exponential in the size of the input clauses), but that it could be presented in a convincing way using only a small amount of space. The reader should imagine that each configuration in Definition 2.3 represents a set of clauses written on a blackboard, where the prover can both write and erase clauses. Our terminology differs from that introduced by Ben-Sasson in [6]; what he describes 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 deriving 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 Games on DAGs

The investigation of RES space is closely associated with the well-known pebbling game and pebbling number of a DAG, originally explored in by Cook and Sethi [11, 10] as a means of investigating bounds on storage requirements.

Definition 2.5 (Pebbling Game on DAGs). *The* pebbling game *is a single-player game played on a DAG G in which the goal is to place a pebble on a designated* target node *of G; we shall assume in what follows that the target is always a unique node of out-degree 0. Vertices of in-degree 0 are* source nodes.

A DAG in which all non-source nodes have in-degree 2 *is called a 'binary DAG'. In the initial position of the game, there are no pebbles on the DAG. Play then proceeds according to one of the following moves:*

1. *The player can place a pebble on an empty source node.*
2. *The player may remove one or more pebbles from nodes.*
3. *For any unpebbled node v, if all of v's immediate predecessors have pebbles on them, then the player may place a pebble on v. Alternatively, the player may choose to slide a pebble from u to v, where u is a predecessor of v.*
4. *The game ends once the target node has a pebble on it.*

Obviously, the player can complete the game simply by placing pebbles successively on all of G's nodes. The game becomes considerably more difficult if the aim is to use as few pebbles as possible.

Definition 2.6 (Pebbling Numbers of DAGs). *The* pebbling number *of G is the minimum number of total pebbles that the player needs in order to complete the pebbling game on G.*

The difficulty of determining the pebbling number of a DAG is emphasized by the following result of Gilbert, Lengauer and Tarjan; this is the basic complexity result on which our main result depends:

Theorem 2.7 ([15]). *Given a DAG G and an integer k, the problem of determining if G can be pebbled with k pebbles is $\mathcal{PSPACE}$-Complete.*

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. Eli Ben-Sasson [6] provides a brief history of how these formulas have been used in the literature.

Definition 2.8 (One-Colour Pebbling Contradictions for DAGs). *The one-colour pebbling contradiction of a DAG G, denoted $Peb(G)$, is a formula constructed from G, consisting of the following clauses:*

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

Each variable x can be interpreted as meaning that the vertex x can be pebbled according to the rules of the pebbling game. This interpretation makes it easy to see that for any DAG G, $Peb(G)$ is unsatisfiable; by imitating the pebbling rules, we can successively deduce all of the one-literal clauses corresponding to the nodes of G, including the target node t, contradicting the initial clause $\{\neg t\}$.

3. Input Resolution, Horn Formulas, and MU Formulas

IRES-UNSAT is defined to be the set of all unsatisfiable formulas that have I-RES refutations, and *URES-UNSAT* the set of all unsatisfiable formulas that have U-RES refutations. 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-SAT*, and the set of all unsatisfiable Horn formulas as *HORN-UNSAT*.

The tree underlying an I-RES refutation has a restricted form that we call a *herringbone*. These are binary trees of depth $d > 0$ in which there are exactly two branches (paths from the root to a leaf) of depth d, and exactly one branch for each depth less than d; for examples, see Figures 3 and 4. As a degenerate case, we also consider the tree with a single node as a herringbone.

A formula F is *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 that 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*.

The pebbling formula $Peb(G)$ is in *HORN-UNSAT*, and in *MU-HORN-UNSAT* for any DAG G in which there is a unique sink node reachable by a directed path from any node in G [18].

3.1 Input Resolution and Unit Resolution

It is well known that every formula in *HORN-UNSAT* has a unit refutation [17]. However, it is easy to see that there are minimally unsatisfiable formulas which have U-RES and I-RES refutations, but are not in *HORN-UNSAT*; for example, the formula $F = (x \vee y) \wedge (\neg x) \wedge (\neg y)$ is not Horn but has an I-RES refutation. That being said, we can characterize formulas in *IRES-UNSAT* by a natural generalization of the class *HORN-UNSAT*.

Define the class *RENAMEABLE-HORN-UNSAT* to be the family of unsatisfiable formulas that are derived from Horn formulas by replacing a literal by its complement. For example, the formula F in the preceding paragraph is in *RENAMEABLE-HORN-UNSAT*, because it can be converted into a formula in *HORN-UNSAT* by the renaming all instances of x and y to their opposite literals.

Theorem 3.1. *Let F be a set of clauses.*

1. *F has an* I-RES *refutation if and only if it has a* U-RES *refutation, which means that IRES-UNSAT = URES-UNSAT.*

2. *If F is minimally unsatisfiable, then F has an input refutation (equivalently, a unit refutation) if and only if F is in RENAMEABLE-HORN-UNSAT.*

Proof: *The first part of the theorem was proved by C.L. Chang [8]. The second part is a result of Henschen and Wos [17].* □

Although the preceding theorem shows that the U-RES and I-RES rules are equivalent as refutation methods, they are not equally efficient, as shown by the next result:

Theorem 3.2. *There is a linear separation between the* I-RES *and* U-RES *proof systems; more specifically,*

1. *For any formula $F \in IRES\text{-}UNSAT$ there exists an* I-RES *refutation of F with size at most $2n+1$, where n is the number of distinct variables in F.*
2. *There exists an infinite family of formulas $F_U \subseteq IRES\text{-}UNSAT$ such that each $F_n \in F_U$ requires* U-RES *refutations of size at least $(n^2 + 3n + 2)/2$, where n is the number of distinct variables in F_n.*

Proof: (Part 1) The proof is by induction on n. For $n = 0$, $F = \{\emptyset\}$, which has an I-RES refutation containing $2n + 1 = 1$ clauses.

Suppose that our statement is true for $n - 1$, and that F is a formula in *IRES-UNSAT* with n variables. Since F has a U-RES refutation, it must contain a unit clause $\{l\}$. Restrict F by setting l to True to produce the formula $F\upharpoonright_{l=True}$, which has $n - 1$ variables, and also has a U-RES refutation. Our induction hypothesis therefore applies, so $F\upharpoonright_{l=True}$ 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 $\{\neg l\}$, where π has size at most $2n-1$. If π is a derivation of the empty clause, then we are done. In the case where π proves $\{\neg l\}$, recall that F contains the input clause $\{l\}$, so simply resolve with this in order to produce the empty clause. In either case we have produced an I-RES refutation containing $\leq 2n - 1 + 2 = 2n + 1$ clauses, as required.

(Part 2): The family F_n is defined inductively as follows:

1. $F_0 = \{\emptyset\}$;
2. $F_{n+1} = \{C \cup \{\neg x_{n+1}\} \mid C \in F_n\} \cup \{\{x_{n+1}\}\}$.

By construction, F_n contains exactly $n + 1$ clauses and 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 $n(n + 1)/2$ clauses derived. In addition, each of the $n+1$ initial clauses must be present in the proof. Any U-RES refutation of F_n therefore contains at least $(n^2 + 3n + 2)/2$ clauses. □

3.2 The Relationship Between Input Resolution and MU Formulas

Having established the relationship between *IRES-UNSAT* and Horn formulas in the previous section, we now investigate the relationship between *IRES-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-IRES-UNSAT*, and unit propagation which will be useful in later sections.

3.2.1 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 3.3 ([1]). *Any minimally unsatisfiable formula on n variables contains at least $n + 1$ clauses.*

This lower bound also has a useful corollary:

Corollary 3.4. *Any 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 3.3 helps us to prove bounds on the size of refutations for minimally unsatisfiable formulas which hold for all forms of Resolution:

Corollary 3.5. *For any minimally unsatisfiable formula F with n variables, any refutation of F in any form of Resolution contains at least $2n + 1$ clauses.*

Proof: This 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 3.3, 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 $2n + 1$. □

Together with the upper bound from Theorem 3.2, this gives us tight bounds on the size of I-RES refutations for *MU-IRES-UNSAT*:

Corollary 3.6. *For any formula $F \in$ MU-IRES-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 3.5, and the upper bound from Theorem 3.2. □

3.2.2 Horn Resolution, Input Resolution and MU(1)

The previous results allow us to relate *MU-IRES-UNSAT* to $MU(1)$:

Theorem 3.7. *Any minimally unsatisfiable formula on n variables which has an* I-RES *refutation contains exactly $n + 1$ clauses, so that $MU\text{-}IRES\text{-}UNSAT \subset MU(1)$.*

Proof: If F is minimally unsatisfiable, and has an I-RES refutation, then by Theorem 3.2, it has an I-RES refutation with $2n+1$ clauses. Of these, n are derived by resolution, so F contains at most $n+1$ clauses. □

In [12], Davydov, Davydova and Kleine-Büning prove a version of Theorem 3.7; their proof is for U-RES rather than I-RES, but Theorem 3.1 shows the equivalence of the results. It is easy to see that the set inclusion in Theorem 3.7 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 Theorems 3.1 and 3.7 allows us to produce a simple set diagram that shows the exact relationship between *IRES-UNSAT*, $MU(1)$, *MU-IRES-UNSAT*, *HORN-UNSAT*, and *MU-HORN-UNSAT*. This diagram is shown below in Figure 1.

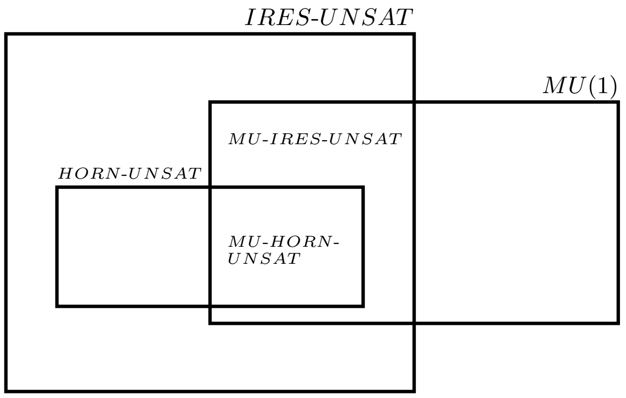

Figure 1. The Relationship Between I-RES, Horn Formulas, and MU(1) Formulas

The exact relationship between I-RES and $MU(1)$ can be illuminated by using a striking characterization of $MU(1)$ due to Oliver Kullmann [20]. Let T be an ordered binary tree (that is to say, the leaves of T are ordered from left to right). Label the interior nodes of T with distinct variables. Now associate a clause C_f with each leaf f in T as follows: Trace down the path from root to f, and for each interior node labeled with a variable x that occurs on the path, include x if the path below the node goes to the left, otherwise include $\neg x$. Define $S(T)$ to be the conjunction of all the clauses C_f, where f is a leaf in T; it is easy to see that $S(T)$ is in $MU(1)$. As an example of this construction, the example given immediately after Theorem 3.7 can be derived from the complete binary tree of depth 2 by an appropriate labeling.

These examples have already occurred in the proof complexity literature. Cook proposed the formulas $S(T)$ based on complete binary trees as examples separating analytic tableaux from T-RES [11]; the general construction appears in published versions of Cook's result [24], [5].

The examples $S(T)$, however, do not encompass all of $MU(1)$. They constitute the *saturated* formulas in $MU(1)$; these formulas have the property that no new literal can be added to a clause without making the formula satisfiable. The construction has to be generalized to produce a characterization of $MU(1)$.

Definition 3.8. *Let T be an ordered binary tree with the interior nodes labeled with distinct variables. A CNF formula F in these variables is* derived from T *if it satisfies the following conditions:*

1. *Each clause in F is associated with a leaf f of T, and is a subset of C_f.*
2. *For every interior node in T labeled with a variable x, the left subtree below x has a leaf with an associated clause containing x, and the right subtree below x has a leaf with an associated clause containing $\neg x$.*

Kullmann's elegant result shows that this construction produces all formulas in $MU(1)$. An earlier related characterization was given by Davydov, Davydova and Kleine-Büning in [12]; they show that $MU(1)$ consists exactly of the formulas which can be encoded as 'basic' matrices. A similar matrix-based characterization of *MU-IRES-UNSAT* can be found in [18].

Theorem 3.9 ([20])**.** *The formulas in $MU(1)$ are exactly those derived from labeled binary trees.*

We can now single out the formulas in *MU-IRES-UNSAT* by choosing an appropriate family of trees.

Theorem 3.10. *The formulas in MU-IRES-UNSAT are exactly those that are based on herringbones.*

Proof: By Theorem 3.2, if F is a formula in *MU-IRES-UNSAT*, then it has an input refutation in which each variable is resolved on exactly once. The tree underlying the refutation is a herringbone. Label the interior nodes of the tree with the variable resolved on at that step in the refutation. Then it is easy to see that F is derived from the resulting labeled tree. □

The preceding characterization also sheds some light on the characterization of *MU-IRES-UNSAT* in Theorem 3.1 due to Henschen and Wos. Horn formulas are exactly those derived from herringbones ordered in such a way that the longest branch occurs on the far right; the renaming operation corresponds to the interchange of left and right subtrees.

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

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

Lemma 3.11. *For any formula F, if F contains two clauses C_1 and C_2 such that $C_1 \subsetneq C_2$, then F is not minimally unsatisfiable.*

Proof: If F is unsatisfiable, and $C_1 \subsetneq C_2$ are both clauses in F, then $F - \{C_2\}$ is also unsatisfiable, since any assignment satisfying C_1 satisfies C_2, so F cannot be minimally unsatisfiable. □

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

Lemma 3.12. *For any formula $F \in MU(1)$, if F contains a unit clause $\{l\}$ for some literal l, then $F\restriction_{l=True} \in MU(1)$.*

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 3.11. Therefore $F\restriction_{l=True}$ contains $n-1$ variables and exactly n clauses, which means that it is minimally unsatisfiable by Corollary 3.4, so $F\restriction_{l=True} \in MU(1)$, as required. □

Combining the previous lemma with Theorem 3.7 yields the following corollary which shows that *MU-IRES-UNSAT* is closed under unit propagation:

Corollary 3.13. *For any formula $F \in$ MU-IRES-UNSAT, if F contains a unit clause $\{l\}$ for some literal l, then $F\restriction_{l=True} \in$ MU-IRES-UNSAT.*

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 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 ([22], 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; hence HORN-UNSAT is $\mathcal{P}$-Complete.*

Jones and Laaser proved that *URES-UNSAT* is $\mathcal{P}$-Complete.

Theorem 4.2 ([19])**.** *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 3.1 that $IRES\text{-}UNSAT = URES\text{-}UNSAT$, this immediately implies the $\mathcal{P}$-Completeness of $IRES\text{-}UNSAT$:

Corollary 4.3. *$IRES\text{-}UNSAT$ is $\mathcal{P}$-Complete, and furthermore there exists an $O(n \cdot m)$ algorithm that 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 $IRES\text{-}UNSAT$ is $\mathcal{P}$-Complete is trivial, since $IRES\text{-}UNSAT = URES\text{-}UNSAT$, which is $\mathcal{P}$-Complete by Theorem 4.2. 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, so that it has a U-RES refutation (and therefore I-RES refutation by Theorem 3.1), or a formula where no more U-RES steps are possible, so that no unit refutation exists. □

In addition, much is also known about the complexity of $MU(k)$:

Theorem 4.4 ([12])**.** *$MU(1) \in \mathcal{P}$ and there exists an $O(n^2)$ algorithm which decides if a formula F is in $MU(1)$, where n is the number of distinct variables in F.*

The preceding theorem has been generalized to all of $MU(k)$ by Fleischner, Kullmann and Szeider; this is the deepest result to date on the structure of MU.

Theorem 4.5 ([20],[14])**.** *For fixed $k > 0$, there is a polynomial-time algorithm to determine whether or not a CNF formula belongs to $MU(k)$.*

4.2 The Tractability of MU-IRES-UNSAT

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

Corollary 4.6. *MU-IRES-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-IRES-UNSAT*. By Corollary 3.4 we know that any formula containing fewer than n variables is not minimally unsatisfiable, so if this is the case, then reject. Similarly, by Theorem 3.7, any formula containing more than $n+1$ variables cannot be in *MU-IRES-UNSAT*, so if this is the case, then reject. Therefore we are left with the case where F contains exactly $n+1$ clauses; we can determine whether or not it has an I-RES refutation (in which case it is in *MU-IRES-UNSAT*) using the $O(n \cdot m)$ algorithm from Corollary 4.3. □

4.3 Tractability of the MU-IRES-UNSAT Size Problem

The size problem for *MU-IRES-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-IRES-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: We can use the algorithm from Corollary 4.6 to determine if $F \in$ *MU-IRES-UNSAT*. If not, then reject, and otherwise compare k with $2n+1$. If $k \geq 2n+1$, then by Corollary 3.6 we can simply accept, and otherwise we reject. □

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

In previous sections we have seen that various versions of the *MU-IRES-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-IRES-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. For $n = 0$, $F = \emptyset$, and so has an I-RES refutation consisting of $\emptyset$. Suppose that our statement is true for $n - 1$, and let F be a formula in $MU(1)$ on n variables that has an I-RES refutation. By Theorem 3.1, F must contain a unit clause $\{l\}$ for some literal l. We want to show that F has an I-RES refutation with top clause C, where C is a in F. In order to apply our induction hypothesis, we restrict F to produce $F' = F\lceil_{l=True}$. By Lemma 3.12, $F\lceil_{l=True} \in MU(1)$. 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 $C\lceil_{l=True} \in F'$ as top clause; let us call this proof π'. There are two cases to consider.

Suppose that $C \neq \{l\}$. Since $F \in MU(1)$ and $\{l\} \in F$ is a unit clause, Lemma 3.11 applies, so the literal l does not occur anywhere else in F, including C. 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.

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 $D = E \cup \{\neg l\}$ 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 $C = \{l\}$. □

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. The 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 showing 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 that can always find proofs that 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 that researchers would like to automatize. However, Alekhnovich and Razborov [4] have shown that, under widely-believed parameterized complexity assumptions, they are not automatizable for the strongest systems (although for Resolution the result is true unless W[P] is in RP, which is a parameterized complexity assumption). 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.

We now prove that for I-RES there exists a polytime algorithm for finding a refutation that 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 that 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 5.2. *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 that 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 4.3 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 the first part of Theorem 3.2 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 NP-Completeness of Input Resolution Size

In previous sections we saw that I-RES is automatizable, *IRES-UNSAT* is $\mathcal{P}$-Complete, that I-RES is optimally automatizable on *MU-IRES-UNSAT*, and the size problem for *MU-IRES-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-UNSAT* is $\mathcal{NP}$-Complete. This result follows immediately from an interesting result proved by Alekhnovich, Buss, Moran and Pitassi [3]:

Theorem 6.1 ([3]). *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-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-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 [3]; it is easy to see that the reduction from Circuit MMSA produces Horn formulas that 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-SIZE*). *IRES-SIZE* $= \{(F, k) \mid F$ *is a formula for which there exists an* I-RES *refutation with size at most* $k\}$

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

Proof: We first show that *IRES-SIZE* is in $\mathcal{NP}$: *IRES-UNSAT* is in $\mathcal{P}$ by Corollary 4.3 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 5.2. 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-SIZE* is $\mathcal{NP}$-Complete via a Cook 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-SIZE*, we can use it as a subroutine to create an algorithm for determining the size of the smallest refutation of F as follows: By Theorem 5.2, 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-SIZE* algorithm to determine the size of the smallest refutation. Since this is only a logarithmic number of subroutine calls, it follows that *IRES-SIZE* is $\mathcal{NP}$-Complete, as required. □

7. The Complexity of Input Resolution Total Space

This section contains a number of results pertaining to pebbling and the total space of I-RES. In Section 7.1 we prove that for any DAG G, the pebbling number of G is equivalent to the total space of any I-RES-W^{-} derivation of a certain variant of the $Peb(G)$ formulas. Following this we show how to dispense with weakening and prove the equivalent result for I-RES. In Section 7.2 we put these equivalence results to use in order to prove the $\mathcal{PSPACE}$-Completeness of the I-RES Total Space Problem. Finally, in Section 8.2 we prove another corollary to our equivalence results, namely an optimal I-RES size/total space tradeoff.

7.1 Equivalence of Pebbling Number and Input Resolution Total Space

We shall prove that for any DAG G, the pebbling number of G is almost exactly equal to the minimum total space of any I-RES-W$^-$ derivation from $Peb(G)^*$ with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ (recall from Definition 2.3 that I-RES-W$^-$ is the I-RES proof system with an added rule of negative weakening).

In analyzing the complexity of I-RES, it is convenient to make a small modification in the definition of the pebbling formulas $Peb(G)$. The new formulas, unlike $Peb(G)$, are in 3-CNF:

Definition 7.1 ($Peb(G)^*$)**.** *For a binary DAG G, the $Peb(G)^*$ formulas are defined in the same way as $Peb(G)$, 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 $\{\neg t, \neg\alpha, \neg\beta\}$.*

In $Peb(G)^*$, there are no positive instances of α or β, so a proof of $Peb(G) \vdash \emptyset$ will correspond exactly with a proof of $Peb(G)^* \vdash \{\neg\alpha, \neg\beta\}$.

If G is a DAG with a unique target node, we define a *pebbling history* $S = S_0, S_1, ..., S_j$ to be a sequence of sets of nodes in G so that $S_0 = \emptyset$ and S_{i+1} is derived from S_i by one of the moves of the pebbling game. A *pebbling strategy* is a history which has met the termination condition of the game (that is to say, the target node of G is in S_j).

Definition 7.2 (I-RES Spine)**.** *Let T be the tree underlying an* I-RES *refutation π of a formula F. The* spine *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 $B = B_0, B_1, ..., B_k$.*

7.1.1 The Equivalence of Pebbling Number and Input Total Space With Weakening

Our first lemma proves the forward direction of our equivalence and states that it is possible to take a pebbling strategy of a DAG G and from it build an I-RES-W$^-$ derivation from $Peb(G)^*$ with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ in which the spine 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 $Peb(G)^* = (4 \vee \neg\alpha \vee \neg\beta) \wedge (5 \vee \neg\alpha \vee \neg\beta) \wedge (6 \vee \neg\alpha \vee \neg\beta) \wedge (\neg 4 \vee \neg 5 \vee 2) \wedge (\neg 5 \vee \neg 6 \vee 3) \wedge (\neg 2 \vee \neg 3 \vee 1) \wedge (\neg 1 \vee \neg\alpha \vee \neg\beta)$.

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

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

Lemma 7.3. *For any DAG G with target node t, if G has a k-pebbling strategy $S = S_0, S_1, S_2 ..., S_{j-1}, S_j$ where $t \in S_j$, then $Peb(G)^*$ 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 spine $B = B_0, B_1, ..., B_{j-1}, B_j$*

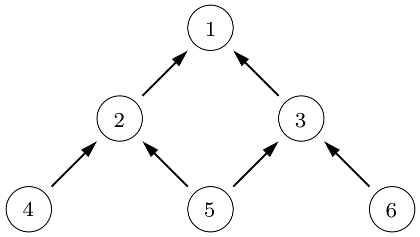

Figure 2. An Example of a Pyramid Graph; The target node is vertex 1.

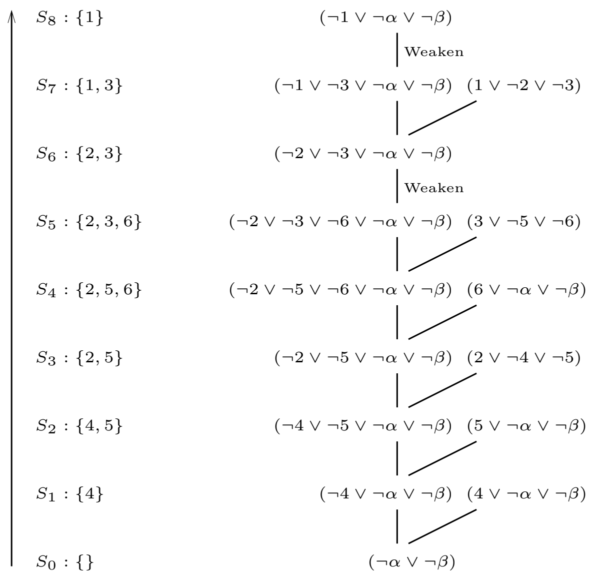
$S_8 : \{1\}$
$(\neg 1 \vee \neg\alpha \vee \neg\beta)$
Weaken
$S_7 : \{1, 3\}$
$(\neg 1 \vee \neg 3 \vee \neg\alpha \vee \neg\beta)$
$(1 \vee \neg 2 \vee \neg 3)$
$S_6 : \{2, 3\}$
$(\neg 2 \vee \neg 3 \vee \neg\alpha \vee \neg\beta)$
Weaken
$S_5 : \{2, 3, 6\}$
$(\neg 2 \vee \neg 3 \vee \neg 6 \vee \neg\alpha \vee \neg\beta)$
$(3 \vee \neg 5 \vee \neg 6)$
$S_4 : \{2, 5, 6\}$
$(\neg 2 \vee \neg 5 \vee \neg 6 \vee \neg\alpha \vee \neg\beta)$
$(6 \vee \neg\alpha \vee \neg\beta)$
$S_3 : \{2, 5\}$
$(\neg 2 \vee \neg 5 \vee \neg\alpha \vee \neg\beta)$
$(2 \vee \neg 4 \vee \neg 5)$
$S_2 : \{4, 5\}$
$(\neg 4 \vee \neg 5 \vee \neg\alpha \vee \neg\beta)$
$(5 \vee \neg\alpha \vee \neg\beta)$
$S_1 : \{4\}$
$(\neg 4 \vee \neg\alpha \vee \neg\beta)$
$(4 \vee \neg\alpha \vee \neg\beta)$
$S_0 : \{\}$
$(\neg\alpha \vee \neg\beta)$

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

(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 $S = S_0, S_1, S_2 ..., S_{j-1}, S_j$ be a k-pebbling strategy 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. Because the dummy literals $\neg\alpha$ and $\neg\beta$ are present in every clause of the proof spine, each spinal clause has a width that is two greater than the corresponding step in the pebbling strategy. Since $S_0 = \emptyset$, and $B_0 = \{\neg\alpha, \neg\beta\} = \emptyset \cup \{\neg\alpha, \neg\beta\}$, the base case of the induction holds.

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 clauses in the spine encode the pebbling strategy, so that $B_i = \bigcup_{v \in S_i} \{\neg v\} \cup \{\neg\alpha, \neg\beta\}$, holds for step S_i. We now show how to translate step $i + 1$ of our pebbling strategy into a corresponding clause in the spine. That is, 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 or pebbles 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 spine such that $B_{i+1} = \bigcup_{v \in S_{i+1}} \{\neg v\} \cup \{\neg\alpha, \neg\beta\}$, so by induction we can translate pebbling into an I-RES-W$^-$ proof in which the spine 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 an I-RES-W$^-$ derivation π and translate it into a pebbling strategy that uses at most two fewer pebbles than π's spinal 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 derivation of from $Peb(G)^*$ with top clause $\{\neg 1, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ into a pebbling strategy $S_0, S_1, ..., S_{10}$ in which the strategy perfectly encodes the spine of the proof. In fact, weakening is never required in this example, so it translates an I-RES rather than an I-RES-W$^-$ derivation to a pebbling strategy. Note how each resolution step may require up to two corresponding pebbling moves.

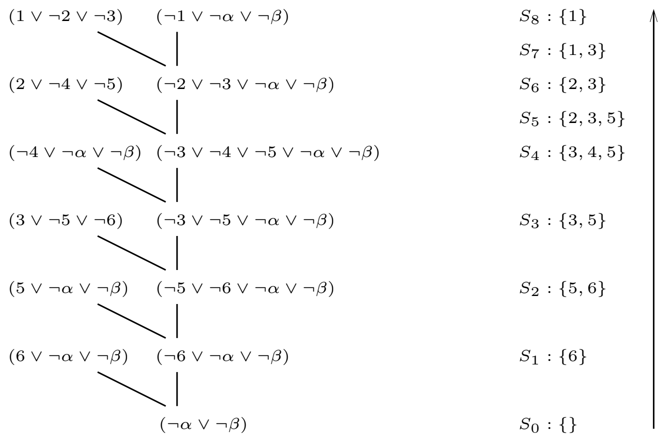
$(1 \vee \neg 2 \vee \neg 3)$
$(\neg 1 \vee \neg\alpha \vee \neg\beta)$
$S_8 : \{1\}$
$S_7 : \{1, 3\}$
$(2 \vee \neg 4 \vee \neg 5)$
$(\neg 2 \vee \neg 3 \vee \neg\alpha \vee \neg\beta)$
$S_6 : \{2, 3\}$
$S_5 : \{2, 3, 5\}$
$(\neg 4 \vee \neg\alpha \vee \neg\beta)$
$(\neg 3 \vee \neg 4 \vee \neg 5 \vee \neg\alpha \vee \neg\beta)$
$S_4 : \{3, 4, 5\}$
$(3 \vee \neg 5 \vee \neg 6)$
$(\neg 3 \vee \neg 5 \vee \neg\alpha \vee \neg\beta)$
$S_3 : \{3, 5\}$
$(5 \vee \neg\alpha \vee \neg\beta)$
$(\neg 5 \vee \neg 6 \vee \neg\alpha \vee \neg\beta)$
$S_2 : \{5, 6\}$
$(6 \vee \neg\alpha \vee \neg\beta)$
$(\neg 6 \vee \neg\alpha \vee \neg\beta)$
$S_1 : \{6\}$
$(\neg\alpha \vee \neg\beta)$
$S_0 : \{\}$

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

Lemma 7.4. *For any DAG G with target node t, if $Peb(G)^*$ 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 a spine $B = B_0, B_1, ..., B_j$ (where B_0 is the goal clause and B_j is the top clause), then there exists a pebbling strategy $S = S_0, S_1, S_2, \ldots, S_l$ where l is $\leq 2j$ with $S_0 = \emptyset$ and $S_l = \{t\}$ 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: We reverse the simulation of Lemma 7.3; there is a slight difference from the previous proof in that we may need to insert some additional pebbling moves as compared to the steps in the I-RES proof. We argue by induction on the length of the spine, starting with the goal clause B_0. The base case is immediate, since $S_0 = \emptyset = B_0 - \{\neg\alpha, \neg\beta\}$.

For the induction step, assume that the simulation has been completed as far as step i, so that $S_q = \bigcup_{\neg v \in B_i}\{v\} - \{\neg\alpha, \neg\beta\}$, where $q \leq 2i$. If the clause B_i is derived from B_{i+1} by resolving with a source clause $\{s, \neg\alpha, \neg\beta\}$ on the variable s, then $B_{i+1} = B_i \cup \{\neg s\}$, so we set $S_{q+1} = S_q \cup \{s\}$; that is to say, we add a pebble to the source node s.

If the clause B_i is derived from B_{i+1} by resolving with a propagation clause $\{\neg a, \neg b, c\}$ on the variable c, then our corresponding pebbling moves depend on whether $\neg a$ or $\neg b$ were already present in B_{i+1}. If exactly one of $\neg a$ or $\neg b$ was already present in B_{i+1} (without loss of generality, let us assume it was $\neg a$), then we slide the pebble on the opposite one (in this case b) in S_q to node c in order to create S_{q+1}. If both $\neg a$ and $\neg b$ were already present in B_{i+1}, then we create S_{q+1} from S_q by placing a pebble on node c. Finally, if neither $\neg a$ nor $\neg b$ were already present in B_{i+1}, then we make two corresponding pebbling moves: We first slide the pebble on a or b to c to create S_{q+1}, and then we remove the other pebble to produce S_{q+2}. In all cases, our pebbling strategy perfectly encodes the spine of the I-RES proof, and never is the number of pebbles too great. □

7.1.2 Dispensing With Weakening

The weakening rule, although useful in showing close links between pebbling strategies and input proofs, is somewhat artificial. In the present section, we show how to remove weakening steps in I-RES-W$^-$ proofs without increasing the size or total space.

Lemma 7.5. *For any unsatisfiable set of Horn clauses F, $C \in F$, if there exists an* I-RES-W$^-$ *proof of D from F with top clause C, with spinal width at most k, then there exists an* I-RES *proof of $D^{'} \subseteq D$ from F with top clause C with spinal width at most k.*

Proof: We argue by induction on the length of the spine of the I-RES-W$^-$ proof of D from F with top clause C. If the proof contains only one clause, C, the Lemma holds trivially. Now assume the Lemma for proofs of length n, and assume given a proof of D with spinal length $n+1$, and spinal width at most k.

If $D = E \cup F$ is derived by weakening from E, then by the induction hypothesis, there is an I-RES proof of $D^{'} \subseteq D$ from F with top clause C, having spinal width at most k. This proof fulfills the condition of the Lemma.

If $D = E \cup F$ is derived from $E \cup \{\neg x\}$ and $F \cup \{x\}$ by resolution, then by the induction hypothesis, there is an I-RES proof of $E^{'} \subseteq E$ from F with top clause C, having spinal width at most k. If $\neg x \notin E'$, then this proof already fulfills the condition of the Lemma; otherwise, resolve E' and $F \cup \{x\}$ to derive $(E' - \{\neg x\}) \cup F \subseteq D$. □

Corollary 7.6. *For any DAG G with target node t, G has a k-pebbling strategy if and only if $Peb^1(G)^*$ has an* I-RES *derivation with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ and goal clause $\{\neg\alpha, \neg\beta\}$ with total space $k+5$.*

Proof: $\Rightarrow$ Assume that there is a k-pebbling strategy for G. By Lemma 7.3, there is an I-RES-W$^-$ proof from $Peb(G)^*$ with top clause $\{\neg t, \neg\alpha, \neg\beta\}$, goal clause $\{\neg\alpha, \neg\beta\}$, and spinal width bounded by $k+2$. It follows from Lemma 7.5 that there is an I-RES proof from $Peb(G)^*$ of a subclause of $\{\neg\alpha, \neg\beta\}$ satisfying the same conditions. However, since the variables α and β occur only negatively in $Peb(G)^*$, they can never be removed from the spine, so the I-RES proof must be a proof of $\{\neg\alpha, \neg\beta\}$.

$\Leftarrow$ Conversely, assume that there is an I-RES proof from $Peb(G)^*$ with top clause $\{\neg t, \neg\alpha, \neg\beta\}$, goal clause $\{\neg\alpha, \neg\beta\}$, and total space at most $k+5$. □

7.2 The PSPACE-Completeness of Input Derivation Total Space

Theorem 7.7. *The following problem is $\mathcal{PSPACE}$-Complete under logspace reducibility: Given a Horn formula F, clauses C, D and integer k as input, does there exist an* I-RES *refutation with top clause C, goal clause D, and total space at most k?*

Proof: We first show that the problem is in $\mathcal{PSPACE}$. We are given an input instance (F, $C \in F$, D, k), we can guess an I-RES refutation of the required sort. 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 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 the problem is in $\mathcal{NPSPACE}$. Finally, we appeal to Savitch's Theorem [23] to show that it is in $\mathcal{PSPACE}$.

Next we show that the problem is $\mathcal{PSPACE}$-Hard by giving a reduction from the $\mathcal{PSPACE}$-Complete pebbling number problem from [15] (See Theorem 2.7). 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 can be pebbled with k pebbles if and only if F has an I-RES derivation with total space at most k of D from top clause C. Our reduction proceeds as follows: Take (G, k) and output $(Peb(G)^*,\ \{\neg t, \neg\alpha, \neg\beta\},\ \{\neg\alpha, \neg\beta\},\ k+5)$, which is clearly a logspace reduction. The proof of correctness for this reduction is given by Corollary 7.6. $\square$

8. Related Complexity Results

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

8.1 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 r-CNF formula, $w(F \vdash_{\text{I-RES}} D) = TS(F \vdash_{\text{I-RES}} D) - r$.

The $Peb(G)^*$ formulas are 3-CNF formulas, so the width of any I-RES derivation of goal clause $\{\neg\alpha, \neg\beta\}$ from $Peb(G)^*$ with top clause $\{\neg t, \neg\alpha, \neg\beta\}$ is $k+5-3 = k+2$. This implies that the I-RES derivation width problem is therefore also $\mathcal{PSPACE}$-Complete:

Corollary 8.1. *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.*

8.2 Optimal Size/Total Space Tradeoffs For Input Resolution

In this section we describe another corollary to the results in Section 7.1. This result also relies closely on one of the most surprising facts concerning pebbling, namely that there exist infinite families of DAGs that 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 [21], where Lingas 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 8.2 ([15])**.** *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 7.6 gives an exact relationship between 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 8.3. *There exists an infinite family of formulas $\mathcal{F} = \{Peb(G)^*|G \in \mathcal{G}\}$ 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 8.2, and let F be any arbitrary formula from $\mathcal{F}$. Therefore $F = Peb(G)^*$ for some $G \in \mathcal{G}$. G can be pebbled with k but not with fewer than k pebbles. By Corollary 7.6, 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 7.6, 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 7.6, there exists a proof π' with total space $k+5+1 = k+6$ and size $O(n)$, as required. □

Acknowledgements

We would like to thank Toni Pitassi for her very helpful comments regarding this paper. In addition, we are grateful for the reports of the referees; these led to considerable improvements in both our results and exposition.

References

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

[2] M. Alekhnovich, E. Ben-Sasson, A.A. Razborov, and A. Wigderson. Space Complexity in Propositional Calculus. *SIAM Journal of Computing*, **31**(4):1184 – 1211, 2001.

[3] 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.

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

[5] N. H. Arai, T. Pitassi, and A. Urquhart. The Complexity of Analytic Tableaux. *Journal of Symbolic Logic*, **71**:777 – 790, 2006. Preliminary version: 33rd ACM Symposium on the Theory of Computing (STOC), 2001, pages 356 - 363.

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

[7] H. Kleine Büning and T. Lettman. *Aussagenlogik: Deduktion und Algorithmen.* B.G. Teubner, Stuttgart, 1994.

[8] C. L. Chang. The Unit Proof and the Input Proof in Theorem Proving. *Journal of the Association for Computing Machinery*, **17**(4):698 – 707, 1970.

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

[10] S. Cook and R. Sethi. Storage Requirements for Deterministic Polynomial Time Recognizable Languages. *Journal of Computer & System Sciences*, **13**(1):25–37, 1976.

[11] S. A. Cook. An Observation on Time-Storage Tradeoff. *Proceedings of the* 5th *Annual ACM Symposium on Theory of Computing (STOC)*, pages 29 – 33, 1973.

[12] 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.

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

[14] H. Fleischner, O. Kullmann, and S. Szeider. Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Theoretical Computer Science*, **289**(1):503 – 516, 2002.

[15] J. R. Gilbert, T. Lengauer, and R. E. Tarjan. The Pebbling Problem is Complete in Polynomial Space. *SIAM Journal of Computing*, **9**(3):513 – 524, 1980.

[16] A. Haken. The Intractability of Resolution. *Theoretical Computer Science*, **39**:297 – 308, 1985.

[17] L. Henschen and L. Wos. Unit Refutations and Horn Sets. *Journal of the Association for Computing Machinery*, **21**:590 – 605, 1974.

[18] A. Hertel. Applications of Games to Propositional Proof Complexity. Ph.D. Thesis, University of Toronto, 2008.

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

[20] O. Kullmann. An Application of Matroid Theory to the SAT Problem. *Proceedings of the* 15th *Annual IEEE Conference on Computational Complexity*, pages 116 – 124, 2000.

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

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

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

[24] A. Urquhart. The Complexity of Propositional Proofs. *The Bulletin of Symbolic Logic*, **1**(4):425 – 467, 1995.