Electronic Colloquium on Computational Complexity, Report No. 46 (2007)

An Exponential Time/Space Speedup For Resolution

Philipp Hertel[*]
Department of Computer Science
University of Toronto
Toronto, ON CANADA
`philipp@cs.toronto.edu`

Toniann Pitassi[†]
Department of Computer Science
University of Toronto
Toronto, ON CANADA
`toni@cs.toronto.edu`

April 20, 2007

Abstract

Satisfiability algorithms have become one of the most practical and successful approaches for solving a variety of real-world problems, including hardware verification, experimental design, planning and diagnosis problems. The main reason for the success is due to highly optimized algorithms for SAT based on resolution. The most successful of these is *clause learning*, a DPLL scheme based on caching intermediate clauses that are "learned" throughout the backtrack search procedure. The main bottleneck to this approach is space, and thus there has been a tremendous amount of research aimed at identifying good heuristics for deciding what information to cache. Haken first suggested a formal approach to this issue, and Ben-Sasson [3] posed the question of whether there is a time/space tradeoff for resolution.

Our main result is an optimal time/space tradeoff for resolution. Namely, we present an infinite family of propositional formulas whose minimal space proofs all have exponential time, but if just three extra units of storage are allowed, then the formulas can be proved in linear time.

We also prove another related theorem. Given an unsatisfiable formula F and an integer k, the resolution space problem is to determine if F has a resolution proof which can be verified using space k. We prove that this problem is PSPACE complete.

[*]Funded by NSERC
[†]Funded by NSERC

ISSN 1433-8092

1 Introduction

The satisfiability problem (SAT) has become a viable and widespread approach for solving real-world problems. SAT procedures are now a standard tool for solving problems in hardware verification, circuit diagnosis, experimental design, planning and diagnosis problems. Surprisingly, the best SAT algorithms are highly optimized variants of DPLL which is nothing more than a backtrack search for a tree-like resolution refutation. The most successful variant, *clause learning*, employs a very clever type of caching scheme. It underlies all state-of-the art complete algorithms for solving SAT. Moreover, clause learning seems capable of solving notoriously hard generalizations of SAT as well, including QSAT [11].

The basic idea behind clause learning is very simple: while performing the backtrack search, store intermediate clauses that are learned along the way, in order to potentially prune the remaining search space. This idea was first suggested in 1973 by Stephen Cook, who referred to it as "Method I" [5]. However it took many years to develop this idea to actually make it work. The main issue stems from the fact that in reality there is only a finite amount of space available. Therefore, all clauses simply cannot be stored, and the difficulty is in obtaining a highly selective and efficient, yet effective caching scheme. This has inspired a great deal of research into methods and heuristics for caching schemes, resulting in state-of-the art algorithms for SAT. [1].

Underlying most of this empirical work is an assumption that there is a *smooth*, nearly linear tradeoff between time and space. For example, *anyspace* algorithms have been developed for SAT and #SAT where a given implementation can use as much space as is currently available [6]. They used empirical results on certain distributions of inputs to suggest that for most ranges of parameters, the tradeoff between runtime and space is nearly linear. In this paper we present theoretical results that run counter to this belief. That is, we demonstrate a family of examples where the runtime (of clause learning algorithms) is linear, but jumps to exponential if the space utilization drops by a constant amount.

As discussed above, while time/space issues for resolution-based satisfiability algorithms has been of central importance for many years, it was only in the late nineties when the formal study of space as a complexity measure for propositional proof systems was initiated. In 1999, Esteban and Toran [7] proposed a definition of space complexity for resolution, called *clause space*, that measures the number of clauses that need to be kept simultaneously in memory in order to verify the resolution refutation. This model is similar to a Turing machine computation with a special read-only input tape from which the axioms can be downloaded from working memory whenever they are needed, and erased from working memory as many times as necessary. This model captures resolution-based SAT algorithms, such as clause learning.

Alekhnovich et al. [2] address the question of how to measure the memory content for more general propositional proof systems. While the most obvious choice is "bit space," [2] introduce the related notion of *variable space*, which counts the number of variable occurrences that must simultaneously be kept in memory. They argue that variable space and bit space are within a logarithmic factor of one another, but variable space makes the model substantially cleaner. Thus we view variable space as the right space measure to study: it applies to a variety of proof systems, and captures in a natural and clean way the space utilization of a broad range of complete algorithms for SAT.

In 2001, Ben-Sasson [4] was the first to study formal time/space tradeoffs for resolution. [1] He asked if there are formulas that have optimal proofs with respect to any one of the parameters, but where optimizing one parameter must cost an increase in the other parameter. He proved that this is the case for tree-like resolution. That is, he showed that there are formulas that have linear tree-like resolution proof size, and that also have constant-sized clause space, but on the other hand, he showed that it is not possible to achieve both (linear size and constant clause space) simultaneously.

However, for general resolution the problem remained open. Our main result is an answer to this ques-

[1]In the algorithms literature, this tradeoff is viewed as a time/space tradeoff, whereas from a proof complexity point of view, the tradeoff would be more accurately called a size/space tradeoff. Size and time are equated because the runtime of a resolution-based SAT algorithm is tightly connected to the size of the underlying resolution proof.

tion, showing that in a very strong sense it is not possible to optimize both size and space simultaneously. We exhibit formulas that require exponential size to refute if restricted to a minimal variable space resolution implementation, but with just three more units of space, the proof size drops to linear! In light of our earlier discussion, this result is surprising, as it runs counter to the belief is that there is a smooth almost linear tradeoff between space and time.

We also prove a related theorem. Given a CNF formula f, the resolution space problem is to determine the minimal-space resolution proof of f. We prove that the resolution space problem is PSPACE complete, affirming that memory management for resolution-based SAT algorithms is a complex issue. The complexity of resolution space is known to be related to the complexity of black-white pebbling games in the following sense [8, 3]. Given a monotone circuit $\mathcal{G}$, a pebbling formula can $Peb(\mathcal{G})$ be associated with $\mathcal{G}$. Intuitively, the associated formula asserts that all of the source vertices are "true"; the sink vertex is "false", and the intermediate gates are consistent with the values of their children. Ben-Sasson proved that from a resolution proof of $Peb(\mathcal{G})$, one can extract a black-white pebbling strategy for $\mathcal{G}$, where the pebbling space is related to the resolution space. Unfortunately the converse is not true in general.

In our work, we construct special formulas $\mathcal{G}$ that allow us to prove a converse relationship. Using this construction we show that the complexity of resolution space is essentially equivalent to the complexity of black-white pebbling of monotone circuits, which we show are both PSPACE-complete. In a related paper [10], the complexity of the classical black-white pebbling problem for *directed acyclic graphs* (dags) is resolved. We note that this result for dags subsumes the PSPACE completeness of black-white pebbling of monotone circuits. However, in order to obtain the resolution space results we need to use a simpler construction that seems possible only for monotone circuits.

2 Overview of Main Results

Essentially all complete algorithms for satisfiability, used in practice, are resolution-based. Let C_1 and C_2 be arbitrary clauses not involving the variable x. Then the resolution rule allows the derivation of the clause $(C_1 \vee C_2)$ from $(C_1 \vee x)$ and $(C_2 \vee \neg x)$ by resolving away x. A resolution proof of a CNF formula f is a sequence of clauses, such that each clause is either a clause from f, or follows from two previous clauses by the resolution rule, and such that the final clause is the empty clause. The *size* of a resolution derivation is the total number of clauses in the proof.

A general implementation of a SAT algorithm based on resolution proceeds by deriving clauses in some way, until eventually either the empty clause is derived, in which case we report *unsatisfiable*, or the algorithm finds a satisfying assignment. Thus the *time* of a resolution-based implementation for SAT is at least the size of the underlying resolution proof produced. The space, intuitively, should be the size of the intermediate clauses that are stored in memory during the process of looking for a resolution proof. In order to formally define space we give the following *configuration-style* definition of a resolution proof from [4].

DEFINITION 2.1 (Configuration-style resolution proof) A configuration $\mathcal{C}$ is a set of clauses. If f is a CNF formula, then the sequence of configurations $\pi = \mathcal{C}[0], \mathcal{C}[1], ..., \mathcal{C}[k]$ is a RES proof of C from F if $\mathcal{C}[0] = \emptyset$, $C \in \mathcal{C}[k]$, and for each $i < k$, $\mathcal{C}[i+1]$ is obtained from $\mathcal{C}[i]$ by one of the following rules: (1) deleting one or more clauses from the current configuration; (2) add the resolvent of two clauses of $\mathcal{C}[i]$; (3) download an axiom (clause) of f. If $\emptyset \in C[k]$, then π is a proof of f.

DEFINITION 2.2: The *variable space* of a proof π is the maximum size of any configuration C in π. The *variable space* of an unsatisfiable CNF formula f is the minimum space over all proofs of f. Unless specified otherwise, space in this paper will refer to variable space. Given a CNF formula f and a number k, the resolution space problem asks whether there is a space k resolution proof of f.

Our first theorem settles the complexity of the resolution space problem. Our second theorem, which is quite surprising, shows that allowing even 3 extra units of storage can have drastic consequences for resolution-based SAT algorithms.

Theorem 1: The resolution space problem is *PSPACE*-complete.

Theorem 2 (Time/Space Tradeoff) There exist CNF formulas such that any minimal space proof of these formulas requires exponential size, but that can be refuted in linear size, with 3 more units of space.

Both of these results are fundamentally tied to generalized black-white pebbling:

DEFINITION 2.3 (Pebbling games on monotone circuits) Let $\mathcal{G}$ be an unbounded-fanin monotone circuit over AND/OR gates, with one distinguished output gate, s. A generalized black-white pebbling strategy for $\mathcal{G}$ is a sequence of pebbling moves. Initially the graph contains no pebbles. At any point a black pebble can be removed from, or a white pebble can be placed on, any node. Further, black pebbles can always be place on and white pebbles can always be removed from source nodes. The goal is to end with a single black pebble on s and no other pebbles on the graph. The rules for pebbling AND/OR nodes are as follows.

1) For any AND-node v, if all of v's predecessors have pebbles on them, then a black pebble can be placed on v, or a black pebble can be slid from a predecessor u to v. For any AND-node v with a white pebble on it, the pebble can be slid to a predecessor u if all other predecessors are pebbled, or the white pebble can be removed if all predecessors are pebbled.

2) For any OR-node v, if at least 1 of a v's predecessors is pebbled, then v can be black pebbled, or a black pebble can be slid from a predecessor u to v. For any OR-node v with a white pebble on it, the white pebble can be slid from v to any predecessor u of v, or the white pebble can be removed if u is pebbled.

Given $\mathcal{G}$ with target node s and a number k, the generalized black-white pebbling problem asks whether there is a pebbling strategy for s which uses at most k pebbles. In order to prove Theorem 1, we will start by proving that generalized black-white pebbling is PSPACE complete, and see how to modify the proof in order to obtain a proof of Theorem 1.

In order to prove the PSPACE completeness of generalized black-white pebbling, we reduce from QSAT. Start with a QBF formula ψ. From ψ, we create a graph $\mathcal{G}$ with the property that ψ is QSAT if and only if $\mathcal{G}$ has a $3n+1$ black-white pebbling strategy. Our construction is similar to [9], where they create a graph from a QBF formula with the property that the formula is QSAT if and only if the graph has a small all-black pebbling strategy. The general idea is to give a small black pebbling of the graph whenever the formula is QSAT, where the pebbling corresponds to the natural (exponential-time but small space) procedure that verifies that ψ is QSAT.

Unfortunately, the graphs used in all earlier constructions are easy to pebble with white pebbles even if the original formula is not QSAT. Thus, we need to alter the graphs in order to impose extra structure soas to prohibit white pebbles from being useful. The idea will be to carefully add extra edges and widgets to the graph so that there will never be any extra “slack”—that is, at almost all times, the number of black pebbles on the graph is at full capacity in order to carry out the upper bound, thereby rendering white pebbles useless if we do not want to exceed our capacity. This will allow us to argue in the lower bound, that no pebbling strategy using only $3n+1$ pebbles can do anything other than the upper bound strategy, even with white pebbles available. Our argument is based on local indegree properties of the graph.

Our reduction and proof will possess two crucial properties. (1) First, the optimal black-white pebbling strategy for $\mathcal{G}$ will be equal to the optimal black pebbling strategy for $\mathcal{G}$. That is, our graph $\mathcal{G}$ will have the important property that white pebbles will not help at all. (2) Secondly, if ψ is QSAT, then any optimal strategy using $3n+1$ pebbles will require exponential time in the number of universal quantifiers. But if we let the strategy use just 3 more pebbles, then the graph $\mathcal{G}$ can be pebbled in linear time, whether or not ψ is QSAT. This result forms the building block for our more complicated results for resolution.

With the above ideas in mind, we will now try to prove the PSPACE completeness of resolution (Theorem 1). Once again we will reduce from QSAT. Start with a QBF formula ψ, and create the graph $\mathcal{G}$ as above. Now from $\mathcal{G}$, we define an associated “pebbling formula”, $Peb(\mathcal{G})$, defined below.

DEFINITION 2.4: (Pebbling formulas)[3] Let $\mathcal{G}$ be a monotone circuit. $Peb(\mathcal{G})$ is a set of clauses, with one variable v_i for each vertex in $\mathcal{G}$, and containing the following (Horn) clauses: (1) For each source vertex v, we have the clause (v); (2) For each AND vertex v with predecessors $u_1,\ldots,u_l$, we have the clauses

$(\neg u_1 \vee \neg u_2 \vee \ldots \vee u_l \vee v)$; (3) Finally for each OR vertex v and each predecessor u of v, we have the clause $(\neg u \vee v)$. By a resolution proof of $Peb(\mathcal{G})$, we mean a resolution derivation of the clause (s) from $Peb(\mathcal{G})$.

As mentioned earlier, [3] showed that from a resolution proof of $Peb(\mathcal{G})$, we can extract a black-white pebbling strategy for $\mathcal{G}$, where the pebbling space is related to resolution space. Unfortunately, the converse does not hold because of the white pebbles; it is not hard to see that from a *pure* black pebbling strategy for $\mathcal{G}$, we do obtain a corresponding space-preserving resolution proof. Now intuitively, since white pebbles do not help for our graphs $\mathcal{G}$, by special property (1) of our reduction, the converse relationship should for our special graphs, even with the addition of white pebbles.

This is the high-level idea behind the argument. However, significant technical difficulties arise when carrying out the proof. The problem comes from the fact that in order to mimic the black pebbling strategy, the axioms need to be downloaded, and this download uses more space (in the resolution simulation) than was needed in the pebbling. This "slack space" causes difficulties in proving the lower bound, since we need to argue that the lower bound still holds even with this extra slack space. This is tricky because it is exactly slack space that allows white pebbles to sneak in and gain advantage over all-black pebbling strategies.

To solve this problem, we further modify $\mathcal{G}$ to obtain $\mathcal{G}_{\mathsf{RES}}$. $\mathcal{G}_{\mathsf{RES}}$ is similar to $\mathcal{G}$ but has even more structure which serves to "fill up" the slack space. Even with this modification, our argument is substantially more complicated than before. We again argue that at each step, the only way for the resolution proof to proceed is to follow the pebbling strategy–however now we will need to use a global graph-theoretic argument, whereas before we essentially argued locally. Our main theorem thus shows that the QBF formula ψ is QSAT if and only if $Peb(\mathcal{G}_{\mathsf{RES}})$ has a space $6n+3$ resolution derivation. Moreover, the above two special properties continue to hold in this context. In particular, our reduction satisfies (2'): If ψ is QSAT, then any space $6n+3$ resolution proof of $Peb(\mathcal{G}_{\mathsf{RES}})$ requires exponential size. However, with $6n+6$ space, for any ψ, the associated formula $Peb(\mathcal{G}_{\mathsf{RES}})$ has a linear-space proof.

Once equipped with Theorem 1 and special property (2'), the proof of Theorem 2 is easy. Start with any totally universally quantified QBF formula $\psi = \forall x_1 \ldots \forall x_n F$, which is QSAT, and corresponding graph $\mathcal{G}_{\mathsf{RES}}$. Since ψ is QSAT, by the properties of our reduction, $Peb(\mathcal{G}_{\mathsf{RES}})$ has a resolution proof of space $6n+3$ which requires exponential size, but there are linear size proofs using $6n+6$ space.

3 The Reductions

QSAT to Circuit $\mathcal{G}$. To show that the generalized black-white pebbling game is PSPACE-complete, we reduce from QSAT. Given a QBF ψ, we produce a monotone circuit $\mathcal{G}$ whose target node s can be black-white pebbled using at most $3n+1$ pebbles if and only if ψ is QSAT. Our monotone circuit is composed of three types of widgets: universal widgets, existential widgets, and clause widgets. The construction of the quantifier widgets relies on a subwidget we call an i-slide, which is designed to severely restrict the player's pebbling strategies. Once the bottom nodes of an i-slide are all black-pebbled, an i-slide strategy, where the bottom pebbles are slid up to the top nodes in the appropriate order, is the only (frugal) way to black-pebble the top nodes without using more than i pebbles. An example of a 4-slide is given in Figure 1.

DEFINITION 3.1: An ***i*-slide** is a pair of sets (V,U) together with a set of edges that satisfy the following properties. V is a set of i nodes $v^1, v^2, \cdots, v^i$ and U is a set of i nodes $u^1, u^2, \cdots, u^i$. The edges are as follows. (1) v^j is the predecessor of all nodes v^k such that $k > j$; (2) u^j is the predecessor of all nodes u^k such that $k > j$; (3) u^j is the predecessor of all nodes v^k such that $k \leq j$; (4) u^j has at least $i-j+1$ predecessors from outside of V or U.

We can now describe our reduction which, given $\psi = Q_n x_n \ldots Q_1 x_1 F$ with m clauses over n variables, outputs $\mathcal{G}$. As in [9], each universally quantified variable is associated with a universal widget and each existentially quantified variable is associated with an existential widget. Each clause of ψ is also associated with a clause widget. These descriptions are meant to be read with the accompaniment of Figures 1, 2, 3, and 4.

The universal widget is depicted in Figure 3. For every i, $1 \le i \le n$, if widget i is a universal widget, it is composed of 4 groups of nodes, $\{\overline{x_i}, \overline{x_i}', d_i, x_i, x_i'\}$, $G_{i-1} = \{g_{i-1}^1, \dots, g_{i-1}^{3i-2}\}$, $\{a_i, b_i\}$, and $G_i = \{g_i^1, \dots, g_i^{3i+1}\}$. These are connected as follows. x_i' has $3i+1$ source nodes $y_{x_i}^1$ through $y_{x_i}^{3i+1}$ as predecessors, d_i has $3i$ source nodes $y_{d_i}^1$ through $y_{d_i}^{3i}$ as predecessors, and $\overline{x_i}'$ has $3i-1$ source nodes $y_{\overline{x_i}}^1$ through $y_{\overline{x_i}}^{3i-1}$ as predecessors. The sole predecessor of x_i is x_i' and the sole predecessor of $\overline{x_i}$ is $\overline{x_i}'$. The subgraph $(\{g_i^1, \dots, g_i^{3i-2}\}, G_{i-1})$ forms an $3i-2$ slide. The node b_i is a successor of every node in G_{i-1}, and the node a_i is a successor of every node in $G_{i-1} \cup \{b_i\}$. Finally, $\overline{x_i}'$ is a predecessor of every node in $\{g_i^1, \dots, g_i^{3i-1}\}$, $\overline{x_i}$ is a predecessor of b_i, d_i is a predecessor of both nodes in $\{b_i, a_i\}$, x_i' is also a predecessor of both nodes in $\{b_i, a_i\}$, x_i is a predecessor of every node in $\{g_i^1, \dots, g_i^{3i}\}$, and a_i is a predecessor of every node in $\{g_i^1, \dots, g_i^{3i+1}\}$. There will be a correspondence between pebbles being placed on x_i', d_i, and $\overline{x_i}$ and an assignment of False to variable x_i, and a correspondence between pebbles being placed on x_i, a_i, and $\overline{x_i}'$ and an assignment of True to variable x_i.

The existential widget is depicted in Figure 4. For every i, $1 \le i \le n$, if widget i is an existential widget, it is composed of 4 groups of nodes, $\{\overline{x_i}, \overline{x_i}', d_i, x_i, x_i'\}$, $G_{i-1} = \{g_{i-1}^1, \dots, g_{i-1}^{3i-2}\}$, $\{f_i, e_i, c_i\}$, and $G_i = \{g_i^1, \dots, g_i^{3i+1}\}$. As in the universal widget, x_i' has $3i+1$ source nodes $y_{x_i}^1$ through $y_{x_i}^{3i+1}$ as predecessors, d_i has $3i$ source nodes $y_{d_i}^1$ through $y_{d_i}^{3i}$ as predecessors, and $\overline{x_i}'$ has $3i-1$ source nodes $y_{\overline{x_i}}^1$ through $y_{\overline{x_i}}^{3i-1}$ as predecessors. The sole predecessor of x_i is x_i' and the sole predecessor of $\overline{x_i}$ is $\overline{x_i}'$. The subgraph $(\{g_i^1, \dots, g_i^{3i-3}\}, \{g_{i-1}^1, \dots, g_{i-1}^{3i-3}\})$ forms a $3i-3$ slide. The node e_i is a successor to every node in G_{i-1}, as is f_i. Also, c_i is an OR-node which is the only successor of e_i and f_i. And every node in $\{g_i^1, \dots, g_i^{3i-2}\}$ is a successor of c_i. Finally, $\overline{x_i}'$ is a predecessor of e_i, $\overline{x_i}$ is a predecessor of f_i as well as every node in $\{g_i^1, \dots, g_i^{3i-1}\}$, d_i is a predecessor of every node in $\{g_i^1, \dots, g_i^{3i}\}$, x_i' is a predecessor of f_i, and x_i is a predecessor of e_i as well as every node in G_i. There will be a correspondence between pebbles being placed on x_i', d_i, and $\overline{x_i}$ and an assignment of False to variable x_i, and a correspondence between pebbles being placed on x_i, d_i, and $\overline{x_i}'$ and an assignment of True to variable x_i.

For every i, $i \ge 1$, G_i is common to widgets i and $i+1$. Also, every member of G_n is a predecessor of the target node s. Each clause widget contains 8 nodes. Consider the clause widget for clause C_j. It contains an OR-node z_j which has 7 predecessors, p_j^{abc}, $abc \in \{0,1\}^3 - \{000\}$, each with indegree 7. The superscript abc indicates one of the "true" settings to the literals underlying the clause. Recall that for each variable x_i of F, there is a group of four associated nodes: x_i, x_i', $\overline{x_i}$, and $\overline{x_i}'$, which are used to encode whether x_i is true or false. The three groups of nodes corresponding to the variables in C_j will be inputs to the p_j^{abc} nodes. Figure 1 shows an example of widget for a clause $(x_1 \vee x_2 \vee x_3)$. The 7^{th} predecessor of each p_j^{abc} is z_{j-1}. Node z_0 is a special source node and is a predecessor of each p_1^{abc}. Finally, the last clause widget is connected to the quantifier widgets via the only node of G_0, g_0^1, which is the same node as z_m.

Construction of $\mathcal{G}_{\mathsf{RES}}$. The graph $\mathcal{G}_{\mathsf{RES}}$ is constructed almost exactly like $\mathcal{G}$ except for the following changes. For every i, $1 \le i \le n$, x_i' has $6i+1$ source nodes $y_{x_i}^1$ through $y_{x_i}^{6i+1}$ as predecessors, d_i has $6i-1$ source nodes $y_{d_i}^1$ through $y_{d_i}^{6i-1}$ as predecessors, and $\overline{x_i}'$ has $6i-3$ source nodes $y_{\overline{x_i}}^1$ through $y_{\overline{x_i}}^{6i-3}$ as predecessors. There is also a target node s which has all $3n+1$ nodes of G_n as predecessors.

We also add new **slack** nodes, each of which is an OR-node, as predecessors to almost every node in the graph. For all i, $1 \le i \le n$, every node v of $G_i \cup \{x_i', \overline{x_i}', d_i\}$ has $3(n-i)$ additional OR-nodes as predecessors, $O(v, x_j)$, $O(v, \overline{x_j})$, and $O(v, d_j)$ for each $i < j \le n$. $O(v, x_j)$ has x_j and x_j' as its sole predecessors, $O(v, \overline{x_k})$ has $\overline{x_j}$ and $\overline{x_j}'$ as its sole predecessors, and $O(v, d_j)$ has d_j and a_j as its sole predecessors if widget j is universal, and has d_j as its sole predecessor if widget j is existential. For all i, $1 \le i \le n$, if widget i is universal, every node v of $\{b_i, a_i\}$ also has $3(n-i)$ addition OR-nodes as predecessors, $O(v, x_j)$, $O(v, \overline{x_j})$, and $O(v, d_j)$ for each $i < j \le n$, which are set up the same way as those for G_i. For all i, $1 \le i \le n$, if widget i is existential, every node v of $\{e_i, f_i\}$ also has $3(n-i)$ addition OR-nodes as predecessors, $O(v, x_j)$, $O(v, \overline{x_j})$, and $O(v, d_j)$ for each $i < j \le n$, which are set up the same way as those for G_i.

Finally, for every j, $1 \le j \le m$, and every $abc \in \{0,1\}^3 - \{000\}$, p_j^{abc} has $3n-6$ additional OR-nodes as predecessors. For each variable x which does not appear in clause j, we include the nodes $O(p_j^{abc}, x)$

and $O(p_j^{abc}, \overline{x})$. And for every i, $1 \leq i \leq n$, we include the node $O(p_j^{abc}, d_i)$. Each of these is set up with predecessors in the same way as the other OR-nodes which were just described.

4 Upper Bounds

Lemma 3: If ψ is QSAT, then the target node s of $\mathcal{G}$ can be pebbled with $3n+1$ pebbles.

DEFINITION 4.1: Let the set of all truth assignments over variables $x_{i+1}, \ldots, x_n$ be denoted by A_i. Thus each α_i in A_i is a partial assignment that sets the outermost $n-i$ variables of $Q_n x_n \ldots Q_1 x_1 F$. For any assignment to α_i, define B_{α_i} to be the pebbling configuration of $\mathcal{G}$ consisting of black pebbles on the following nodes: For each universally quantified variable x_j, $j \geq i+1$ if $\alpha_i(x_j) = 0$, then $x_j' \in B_{\alpha_i}$, $d_j \in B_{\alpha_i}$, and $(\overline{x_j}, \overline{x_j'}) \in B_{\alpha_i}$. Otherwise, if $\alpha_i(x_j) = 1$, then $\overline{x_j'} \in B_{\alpha_i}$, $a_j \in B_{\alpha_i}$ and $(x_j, x_j') \in B_{\alpha_i}$. The pebbling configurations for any existentially quantified variable x_j, $j \geq i+1$ are defined almost identically except that if $\alpha_i(x_j) = 1$, then $\overline{x_j'} \in B_{\alpha_i}$, $d_j \in B_{\alpha_i}$ and $(x_j, x_j') \in B_{\alpha_i}$.

DEFINITION 4.2 (*Black clamping interval)* Let $t_0 \leq t_j \leq t_k \leq t_{end}$. Let S be a set of nodes. We say that $S \in [t_a, t_b]$ if all nodes from S must be black pebbled during every configuration from time t_a through time t_b. We say that $(u, v) \in [t_a, t_b]$ if either u or v is black pebbled during every configuration from time t_a to time t_b.

Lemma 3 follows from the following more general lemma by setting $i = n$.

Lemma 4: For all i, $\alpha_i \in A_i$, suppose the graph $\mathcal{G}$ is initially in configuration B_{α_i}. If ψ is QSAT, then we can black pebble G_i at some time $t > 1$ using $3n+1$ pebbles, while keeping B_{α_i} clamped (i.e., $B_{\alpha_i} \in [1, t]$.)

Proof: The proof is by induction on i from 0 to n. The base case is when $i = 0$. Let α_0 be any assignment in A_0. Suppose that $Q_n x_n \cdots Q_1 x_1 F\lceil_{\alpha_0}$ is QSAT. Then clearly $F\lceil_{\alpha_0} = 1$, so some literal in every clause must be set to true. This implies that for each z_j, $1 \leq j \leq m$, all the predecessors except z_{j-1} of some p_j^{abc} are all black pebbled in C_{α_0}. We can therefore black pebble G_0 as follows. Place a black pebble on the source node z_0. This pebbles the final unpebbled predecessor of some p_1^{abc}. We can therefore slide the black pebble from z_0 to p_1^{abc} and then to z_1. This allows us to pebble z_2 in the same way. Inductively, it is clear that we can slide the black pebble all the way to z_m which is g_0^1. Note that this strategy uses only black pebbles. We now prove the induction step in which we will show that if $\psi\lceil_{\alpha_i}$ is QSAT, then we can black pebble $G_i = \{g_i^1 \cdots g_i^{3i+1}\}$ using no more than $3i+1$ pebbles without moving any pebbles in B_{α_i}.

Case 1: Q_i is a universal quantifier. In this case, both $\psi\lceil_{\alpha_i \cup \{x_i\}}$ and $\psi\lceil_{\alpha_i \cup \{\overline{x_i}\}}$ are QSAT. We begin in configuration B_{α_i} with $3i+1$ free pebbles. Black pebble x_i', followed by d_i, and then $\overline{x_i}'$. Then move the pebble from $\overline{x_i}'$ to $\overline{x_i}$. At this point we have $3i-2$ pebbles free and can apply the induction hypothesis to black pebble G_{i-1}. Then slide the black pebble from $\overline{x_i}$ to b_i, then the black pebble from d_i to a_i. Remove all pebbles from widget i except for the ones on a_i and x_i'. Then slide the black pebble from x_i' to x_i and black pebble $\overline{x_i'}$ again. Now apply the induction hypothesis to simultaneously black pebble G_{i-1} again. Next, use the i-slide strategy to slide all of G_{i-1}'s pebbles up to g_i^1 to g_i^{3i-2}. Then slide $\overline{x_i'}$'s black pebble to g_i^{3i-1}, and then x_i's black pebble to g_i^{3i}. Finally, slide the black pebble from a_i to g_i^{3i+1}.

Case 2: Q_i is an existential quantifier. In this case, either $\psi\lceil_{\alpha_i \cup \{x_i\}}$ or $\psi\lceil_{\alpha_i \cup \{\overline{x_i}\}}$ is QSAT. As in the universal case, we begin in B_{α_i} with $3i+1$ free pebbles. Black pebble x_i', followed by d_i, and then $\overline{x_i}'$. If $\psi\lceil_{\alpha_i \cup \{x_i\}}$ is QSAT, move the black pebble from x_i' to x_i. Then apply the induction hypothesis to black pebble G_{i-1}. Next, slide the black pebble from g_{i-1}^{3i-2} to e_i and then to c_i. Then move the black pebble from $\overline{x_i'}$ to $\overline{x_i}$. If $\psi\lceil_{\alpha_i \cup \{x_i\}}$ is not QSAT and $\psi\lceil_{\alpha_i \cup \{\overline{x_i}\}}$ is, move the black pebble from $\overline{x_i'}$ to $\overline{x_i}$. Then apply the induction hypothesis to black pebble G_{i-1}. Next, slide the black pebble from g_{i-1}^{3i-2} to f_i and then to c_i. Then move the black pebble from x_i' to x_i. In either case there are now black pebbles on g_{i-1}^1 to g_{i-1}^{3i-3}, c_i, x_i, d_i, and $\overline{x_i}$. Use the i-slide strategy to slide all the black pebbles on g_{i-1}^1 through g_{i-1}^{3i-3} to g_i^1 through g_i^{3i-3}. Then slide the black pebble from c_i to g_i^{3i-2}, followed by the black pebble from $\overline{x_i}$ to g_i^{3i-1}, and then the black pebble from d_i to g_i^{3i}. Finish by sliding the black pebble from x_i to g_i^{3i+1}. $\square$

Lemma 5: There is a black pebbling strategy for $\mathcal{G}$ which pebbles s in time $O(|\mathcal{G}|)$ and uses $3n+4$ pebbles, regardless of whether ψ is QSAT.

Proof: Begin by pebbling x_i', d_i, $\overline{x_i'}$ from $i = n$ down to i, thereby assigning "double" false to every variable. This uses $3n$ space. We can now pebble G_0 using only 4 more pebbles. Begin by pebbling the source node z_0. Then place a pebble on the positive node of one of the literals in clause 1. If this literal is $\overline{x_j}$, then we now have black pebbles on x_j', d_j, $\overline{x_j}$, and $\overline{x_j'}$. Do this to each literal of the clause. This uses 3 pebbles so we have $3n+4$ pebbles on the circuit, equaling the space bound. But one can now pebble p_1^{111} by sliding the pebble from z_0 to it. Then slide the pebble from p_1^{111} to z_1. Pick up the three extra black pebbles which were placed on the positive literals' nodes. At this point there are $3n+1$ pebbles on the circuit, $3n$ to assign double false to each variable and a black pebble on z_1. Continue this process inductively until $3n$ pebbles assign double false to each variable and one pebble is on $z_m = g_0^1$.

We must now pebble through each quantifier widget. We can pebble through each existential widget as in Lemma 3. We require an extra pebble for the universal widgets, to avoid black pebbling G_{i-1} a second time. To do so, put the extra pebble on $\overline{x_i}$ right after pebbling G_{i-1} to have black pebbles on G_{i-1}, d_i, x_i', and both $\overline{x_i'}$ and $\overline{x_i}$. Now pebble b_i and a_i sliding pebbles from $\overline{x_i}$ to b_i and then d_i to a_i. Remove the extra pebble from b_i. At this point, the maximum number of pebbles used in the widget is $3n+2$ right before a_i was pebbled, and there are black pebbles on every node of G_{i-1}, $\overline{x_i'}$, a_i, and x_i'. Slide the pebble from x_i' to x_i and continue as in Lemma 3. Since we only have to pebble each clause widget and each quantifier widget once, the whole process takes $O(n+m)$ time and uses no more than $3n+4$ pebbles. $\square$

Finally, we prove the pebbling strategies given above for $\mathcal{G}$ can be mimicked in resolution, to give small space resolution derivations for $Peb(\mathcal{G}_{\mathsf{RES}})$.

Lemma 6: The k-pebble, t-time black pebbling strategies of Lemmas 3 & 5 for the target node s of $\mathcal{G}'$ imply the existence of space $k+d$ resolution derivations of $pu(G_n)$ in $Peb(\mathcal{G}_{\mathsf{RES}})$ which take time polynomial in t, where d is the maximum size of any axiom of $Peb(\mathcal{G}_{\mathsf{RES}})$.

Proof: We will derive the positive unit clauses for each variable of $\mathcal{G}'$ in the same order as the nodes of $\mathcal{G}$ are black pebbled, and remove positive unit clauses from memory when the corresponding black pebbles are removed. Simulating the slide move requires a little attention. Suppose a black pebble is slid from node u to v in widget i. In this case we want to derive $pu(v)$ and remove $pu(u)$ from memory. We first download $ax(v)$. This is where we encounter the maximum space. We then resolve the positive unit clauses of v's predecessors with $ax(v)$ and then remove $pu(u)$. Note that whenever a node is first black pebbled, all of its assignment variables are clamped. This means that their positive unit clauses are in our current resolution configuration. So we can follow the removal of $pu(u)$ by downloading, for each $j > i$, the appropriate axiom for each variable $O(v, x_j)$, $O(v, d_j)$, $O(v, \overline{x_j})$ and resolve each with the corresponding positive unit clause, and then resolve the result with $ax(v)$. This will yield $pu(v)$, and we will once again be in a state in which the resolution configuration corresponds to the black pebbling configuration. $\square$

5 Lower Bounds

In this section we will prove the following theorem, which shows that generalized black-white pebbling is PSPACE complete. We present this result first since it is simpler, and forms the basis for the analogous result for resolution (Theorem 10). The proof of Theorem 10 follows.

Theorem 7: Let ψ be a QBF, and let $\mathcal{G}$ be the corresponding monotone circuit. If $\mathcal{G}$ has a $3n+1$ pebbling, then ψ is QSAT, and any pebbling strategy using $3n+1$ pebbles requires $\Omega(2^k)$ steps, where k is the number of universal quantifiers in ψ.

The above theorem follows from the following more general theorem.

Theorem 8: For all $\alpha_i \in A_i$, if there exists times t', t'' such that $B_{\alpha_i} \subseteq [t', t'']$, then black pebbling G_i at t'' from B_{α_i} using no more than $3n+1$ pebbles, requires that ψ is QSAT and requires $\Omega(2^k)$ units of time between t' and t'', where k is the number of universal quantifiers among the i inner most quantifiers.

The following lemma will be used repeatedly. In particular, it implies that for any i-slide (V,U), in order to pebble V using no more than i pebbles, U must first be black pebbled at some earlier time.

Lemma 9: If a node v has k predecessors and there are $3n+1-k$ other nodes in $[t',t'']$ and v is not white pebbled at t'', then v can be black pebbled at most once and can never be white pebbled between t' and t''.

Proof: If v is white pebbled, then its white pebble can only be discharged once it has contributed toward placing a black pebble beyond it. The existence of this extra black pebble means that there are at most $k-1$ free pebbles to pebble all of v's k predecessors. So the space bound must be exceeded to discharge the white pebble. The same argument forbids a second black pebbling. $\square$

Proof: [**of Theorem 8**] The proof is by induction on i from 0 to n. The base case is when $i=0$. Let α_0 be any assignment in A_0 and suppose there exist times t' and t'' such that $B_{\alpha_0} \subseteq [t',t'']$. We will show that simultaneously black pebbling G_0 at t'' without ever exceeding $3n+1$ pebbles requires that ψ is QSAT.Suppose for sake of contradiction that $\psi\lceil_{\alpha_0}$ is not QSAT. In order to black pebble G_0 using 1 pebble, we can clearly never use a white pebble, since by frugality we will have to place a black beyond it while it is still on the circuit, which will require at least 2 pebbles. Therefore, it is clear that the black pebble on G_0 must be slid up from z_m. Also, it is clear that the black pebble must be slid to z_m from some p_m^{abc}. Since, B_{α_0} does not include z_{m-1}, the black pebble must have been used to pebble z_{m-1}. Inductively, we can see that the black must slide through each clause widget, and must have first been placed on the circuit at z_0. But if $F\lceil_{\alpha_0}$ is not QSAT then there is some clause node z_j such that some predecessor of each p_j^{abc}, other than z_{j-1}, is unpebbled by B_{α_0}. Then pebbling z_j will require two pebbles, one on z_{j-1} and one on the unpebbled predecessor of some p_j^{abc}. Since we only have 1 pebble at our disposal, this is impossible. For the inductive step, there are two cases. Either the i^{th} quantifier is universal or existential.

Case 1: Q_i is a universal quantifier.We will show that in order to black pebble G_i we must necessarily pass through a number of all-black configurations, including black pebbling G_{i-1} twice, once with black pebbles on x_i', d_i, and either $\overline{x_i}$ or $\overline{x_i'}$ (the false configuration), and once with black pebbles on $\overline{x_i'}$, a_i, and either x_i or x_{n-i+1}' (the true configuration). We prove this by pointing out 14 important partial configurations B_0 to B_{13}, which occur at times t_0 to t_{13} respectively, where B_0 is B_{α_i} and B_{13} has G_i black pebbled.

We appeal to Lemma 9 to conclude that no node in $G_i \cup \{a_i, b_i, x_i'\}$ can be white pebbled and each can only be black pebbled once between t_0 and t_{13}. Since x_i' has $3i+1$ source nodes as predecessors, our first action within widget i must be to black pebble x_i' and it must stay in place until its successor x_i' is pebbled for the last time. Then a pebble must remain on x_i until all of its successors are pebbled for the last time, because we can never repebble/discharge x_i once x_i' is empty. Let t_6 be the time that a_i is pebbled and let t_{11} be the time g_i^{3i} is pebbled. Then $x_i' \in [t_1, t_6-1]$ and $(x_i, x_i') \in [t_6, t_{11}-1]$.

Our argument now divides into two sections. In order to simultaneously black pebble G_i we must black pebble g_i^{3i+1}, which requires that both a_i and $\{g_i^1, \dots, g_i^{3i}\}$ be pebbled. In the first part of the argument we prove that in order to black pebble a_i, $\psi\lceil_{\alpha_i \cup \{\bar{x_i}\}}$ must be QSAT and that $\Omega(2^k)$ units of time must pass between t_0 and t_6, where k is the number of universally quantified variables among the inner most $i-1$ variables of ψ. In the second part of the argument, we argue that $g_i^1, \dots, g_i^{3i}$ must also be simultaneously black pebbled in order to black pebble g_i^{3i+1} and that pebbling them without exceeding our bound necessitates that $\psi\lceil_{\alpha_i \cup \{x_i\}}$ is QSAT and that $\Omega(2^k)$ units of time pass between times t_6 and $t_{13}-1$. This will allow us to conclude that black pebbling G_i requires that $\psi\lceil_{\alpha_i}$ is QSAT and requires $\Omega(2^{k'})$ time, where $k' = k+1$ is the number of universally quantified variables among the inner most i variables of ψ.

Since a_i can only be black pebbled once and is needed to pebble each node of G_i, $a_i \in [t_6, t_{13}-1]$. In order to black pebble a_i at time t_6 we must pebble b_i at some time t_5, before t_6. Again, we know that b_i can only be black pebbled once in t_0 to t_{13}, so $b_i \in [t_5, t_6-1]$. Also, d_i is a predecessor of both a_i and b_i and must be pebbled at times t_5-1 and t_6-1. Since x_i' is in $[t_1, t_6]$, by Lemma 9 we can conclude that d_i cannot be white pebbled and can only be black pebbled once in this interval. Also, since it has in-degree $3i$, d_i must be black pebbled at t_2, immediately after t_1 as in Lemma 3, so $d_i \in [t_2, t_6-1]$. The same argument can be made to argue that $(\overline{x_i}, \overline{x_i'}) \in [t_3, t_5 - 1]$, where t_3 is after t_2. In order to black pebble a_i or b_i, we must first pebble G_{i-1} at some time t_4 before t_5. This whole time the nodes x_i', d_i, and $(\overline{x_i}, \overline{x_i'})$ are clamped. We can therefore apply Lemma 9 to conclude that G_{i-1} must be black pebbled at some time t_4 between t_3 and t_5 . We can now apply the induction hypothesis to conclude that black pebbling G_{i-1} requires $\psi\lceil_{\alpha_i \cup \{\overline{x_i}\}}$ to be QSAT and black pebbling G_{i-1} from $B[t_3]$ requires time $\Omega(2^k)$, where k is the number of universally quantified variables among the inner most $i-1$ variables of ψ.

We now proceed with the second phase of the argument. We know that each node in G_i cannot be white pebbled and can only be black pebbled once. So when we black pebble g_i^{3i+1} at time t_{13}, all the rest of G_i must already be black pebbled. Consider g_i^{3i}. In order to black pebble it at time t_{12} before t_{13}, we must first black pebble g_{i-1}^{3i} at time t_{11}. In order to black pebble g_{i-1}^{3i} at time t_{11} we must first black pebble $g_i^1, \ldots, g_i^{3i-2}$. But we must also pebble $\overline{x_i'}$. Note that $\overline{x_i'}$ must be empty at t_6 since a_i has $3i+1$ predecessors, none of which is $\overline{x_i'}$. Also, $\overline{x_i'}$ must be empty again by $t_{12}-1$, since g_i^{3i} has $3i+1$ predecessors, none of which is $\overline{x_i'}$. We can therefore apply Lemma 9 to conclude that between t_6 and t_{13}, $\overline{x_i'}$ cannot be white pebbled and can only be black pebbled once. We must therefore repebble $\overline{x_i'}$ at some time t_7 after t_6 when a_i and (x_i, x_i') are clamped and $\overline{x_i'} \in [t_7, t_{11} - 1]$. Since $\overline{x_i'}$ is a predecessor of every node in $g_i^1, \ldots, g_i^{3i-2}$, these nodes can only be black pebbled at some time t_{10}, with g_i^1 being pebbled first at t_9, after t_7. Every node of G_{i-1} is a predecessor of g_i^1. Since the three nodes $\{\overline{x_i'}, a_i, (x_i, x_i')\} \subseteq [t_6, t_{10}]$ we can apply Lemma 9 to conclude that G_{i-1} must be black pebbled at t_8 between t_7 and t_9. Since $\{\overline{x_i'}, a_i, (x_i, x_i')\}$ is the true assignment for variable x_i we can apply our induction hypothesis to conclude that $\psi\lceil_{\alpha_i \cup \{x_i\}}$ must be QSAT and black pebbling G_{i-1} from $B[t_7]$ requires time 2^k, where k is the number of universally quantified variables among the inner most $i-1$ variables of ψ.

Thus we have shown that any $3n+1$ pebbling must black pebble G_{i-1} twice between t_0 and t_{15}, once implying that $\psi\lceil_{\alpha_i \cup \{\overline{x_i}\}}$ is QSAT, and once implying that $\psi\lceil_{\alpha_i \cup \{x_i\}}$ is QSAT. Each time requires $\Omega(2^k)$ time, where k is the number of universally quantified variables among the inner most $i-1$ variables of ψ. Therefore, black pebbling G_i requires time $\Omega(2^{k+1})$, and implies that $\psi\lceil_{\alpha_i}$ is QSAT.

Case 2: Q_i is an existential quantifier. We prove this by pointing out 12 important partial configurations $B[t_0]$ to $B[t_{11}]$, which occur at times t_0 to t_{11} respectively, where $B[t_0]$ is B_{α_i} and $B[t_{11}]$ has G_i black pebbled.

By Lemma 9, no node in G_i can be white pebbled between t_0 and t_{11}, and each can be black pebbled at most once. Based on which nodes of G_i are predecessors to others, we can conclude that g_i^{3i+1} must be black pebbled last, at time t_{11}, g_i^{3i} must be black pebbled before that at time t_{10}, g_i^{3i-1} must be pebbled before that at time t_9, g_i^{3i-2} must be pebbled before that at time t_8, and g_i^{3i-3} must be black pebbled at some time t_7, before t_8. In order to do this, g_i^1 must be black pebbled at some time t_6, before t_7. And since each can only be pebbled once, they are all clamped through t_{11}.

We now argue that x_i must be black pebbled at $t_{11}-1$. All of g_i^{3i+1}'s $3i$ other predecessors are clamped until t_{11}, so g_i^{3i+1} must receive its black pebble by sliding it from its single other predecessor x_i. Furthermore, we can apply Lemma 9 to show that x_i's predecessor, x_i', cannot be white pebbled and can only be black pebbled once between t_0 and t_{11}. Since x_i' has $3i+1$ source nodes as predecessors, our first action within widget i must be to black pebble x_i' and it must stay in place until all of its successors are pebbled for the last time. Then a pebble must remain on x_i until all of its successors are pebbled for the last time, because we can never repebble/discharge x_i once x_i' is empty. So $(x_i, x_i') \in [t_1, t_{11} - 1]$.

Once we know that at least one node of x_i and x_i' is pebbled throughout the time period, we can use a similar argument to show that $d_i \in [t_2, t_{10} - 1]$, where t_2 comes after t_1, and then, once at least two nodes are clamped $(\overline{x_i}, \overline{x_i'}) \in [t_3, t_9 - 1]$, where t_3 comes after t_2. Also, by Lemma 9, neither e_i nor f_i can be white pebbled between t_0 and t_{11} and each can be black pebbled at most once between t_0 and t_{11}.

We now consider g_i^1 to g_i^{3i-2}, which require that the OR-node c_i is pebbled. We know that c_i must be black pebbled because g_i^{3i-2} has in-degree $3i+1$, and all of its other predecessors are clamped. Therefore, it must have received its black pebble by sliding it from c_i. So c_i must hold a black pebble at some time t_5, before t_8. We can also prove that t_5 occurs before t_6. We know this because g_i^1 must be black pebbled at t_6 and at $t_6 - 1$ all $3i+1$ of g_i^1's predecessors must be black pebbled. This means that the rest of the widget, including f_i and e_i must be empty. But we already know that they can only be black pebbled once between

t_0 and t_{11} and that had to have occurred at some time t_4, before t_5. This means that we cannot repebble c_i after g_i^1 is pebbled, because it requires either e_i or f_i to be pebbled. Therefore, $c_i \in [t_5, t_8 - 1]$.

At t_4 either e_i or f_i is black pebbled. If e_i is black pebbled at t_4, then $G_{i-1} \cup \{x_i, d_i, \overline{x_i'}\}$ must be pebbled at time $t_4 - 1$. If f_i is black pebbled at t_4, then $G_{i-1} \cup \{x_i', d_i, \overline{x_i}\}$ must be pebbled at time $t_4 - 1$. We now want to prove that in either case, G_{i-1} must be simultaneously black pebbled at $t_4 - 1$. To do this, note that widget i always contains at least 4 pebbles from t_4 until t_{11}. This means that after t_4 we cannot black pebble or discharge a white pebble from any node G_{i-1} since they all have in-degree $3i - 2$. This means that g_{i-1}^1 to g_{i-1}^{3i-3} must all be in $[t_4 - 1, t_6]$ since they are needed again to black pebble g_i^1. This means that the only one of either e_i or f_i's $3i + 1$ predecessors which is unclamped at $t_4 - 1$ is g_{i-1}^{3i-2}. Therefore either e_i or f_i must have received its black pebble from g_{i-1}^{3i-2} via a slide move. The rest of G_{i-1} must also be black because g_{i-1}^1 to g_{i-1}^{3i-3} form a $3i-3$-slide with g_i^1 to g_i^{3i-3} when $3n + 1 - (3i - 3)$ nodes are unclamped between t_5 and t_6. So by Lemma 9 and the fact that they can never be repebbled between t_4 and t_{11}, g_{i-1}^1 to g_{i-1}^{3i-3} must be simultaneously black pebbled at $t_4 - 1$. Therefore, if e_i is black pebbled at t_4, then we can apply the induction hypothesis to conclude that $\psi\lceil_{\alpha_i \cup \{x_i\}}$ is QSAT and that simultaneously black pebbling G_{i-1} from $B[t_3]$ required time 2^k, where k is the number of universally quantified variables among the inner most $i - 1$ variables of F. And if f_i is black pebbled at t_4, then we can apply the induction hypothesis to conclude that $\psi\lceil_{\alpha_i \cup \{\overline{x_i}\}}$ is QSAT and simultaneously black pebbling G_{i-1} from $B[t_3]$ required time 2^k, where k is the number of universally quantified variables among the inner most $i - 1$ variables of F. In either case, $\psi\lceil_{\alpha_i}$ is QSAT and simultaneously black pebbling G_{i-1} from $B[t_3]$ requires time 2^k, where k is the number of universally quantified variables among the inner most i variables of ψ. $\square$

Theorem 10: Let ψ be a QBF, and let $\mathcal{G}_{RES}$ be the associated graph. Then if $pu(s)$ can be derived from $Peb(\mathcal{G}_{RES})$ using no more than $6n + 3$ space, then ψ is QSAT, and any $6n + 3$ space proof requires $\Omega(2^k)$ steps, where k is the number of universally quantified variables in ψ.

The rest of this section is devoted to the proof of the above theorem. We begin with some definitions.

DEFINITION 5.1: We partition some of the nodes and their associated variables into three sets, **assignment** nodes and variables, **entry point** nodes and variables, and **internal** nodes and variables. We refer to the nodes in $\bigcup_{i=1}^n \{x_i, x_i', \overline{x_i}, \overline{x_i'}, a_i, d_i\}$ as assignment nodes, and the associated variables as assignment variables. We refer to the nodes in $\{g_0^1\} \cup \bigcup_{i=1}^n \{x_i', \overline{x_i'}, d_i\}$ as entry point nodes, and the associated variables as entry point variables. For universal widgets, we refer to $\bigcup_{i=1}^n (G_i \cup \{a_i, b_i\})$ as internal nodes, and the associated variables are internal variables. And for existential widgets, we refer to $\bigcup_{i=1}^n (G_i \cup \{e_i, f_i\})$ as internal nodes. The target node s is also an internal node of $\mathcal{G}$.

Each entry point node v has a dual, $dual(v)$, where $dual(g_0^1) = g_0^1$, and for each i, $1 \le i \le n$, $dual(\overline{x_i'}) = \overline{x_i}$, and $dual(x_i') = x_i$. If widget i is universal, then $dual(d_i) = a_i$ and if it is existential then $dual(d_i) = d_i$. The initial clause of $Peb(\mathcal{G})$ containing x positively is called $ax(x)$, and the positive unit clause for x is $pu(x)$. We refer to the set of clauses containing x positively as $pos(x)$. For any set of variables X, $pu(X) = \{pu(x) \mid x \in X\}$.

DEFINITION 5.2: A **clamping interval** $[t_j, t_k]$ is the set of clauses which must be in memory at all times between t_j and t_k inclusive. Let C_1, C_2 be two clauses. We say that $(C_1, C_2) \in [t_j, t_k]$ if either C_1 or C_2 is in memory during every configuration from time t_j through time t_k inclusive. We say that $pos(x) \in [t_j, t_k]$ if for each configuration $\mathcal{C}[t']$, $t_j \le t' \le t_k$, some clause of $pos(x)$ is in $\mathcal{C}[t']$. We say that **an assignment variable** x **is clamped** from configuration $\mathcal{C}[t_j]$ to configuration $\mathcal{C}[t_k]$ if either $pos(x) \in [t_j, t_k]$ or $pos(dual(x)) \in [t_j, t_k]$. We say that a node v's **assignment variables are clamped** within an interval, if for every one of v's slack nodes $O(v, z)$, the assignment variable z is clamped in that interval.

DEFINITION 5.3: We say that a node z is **downstream** of another node y if there is a path from y to z. We say that a node z is downstream of another node y **in a path** ρ if there a subpath of ρ from node y to node z.

We say that a node y is **upstream** of another node z if z is downstream of y and we say that y is upstream of z in a path ρ if z is downstream of y in ρ.

DEFINITION 5.4: Let ρ be a path from node y to node z. We say that a clause C **blocks** ρ at node b, if $b \in \rho$, $b \neq y$ and the variable associated with b appears positively in C and no variable associated with any node on the subpath of ρ from y to b appears negatively in C. If the current resolution configuration contains a clause which blocks ρ at some node, then we say that ρ is **blocked**. Otherwise we say that ρ is **unblocked**. For a given path ρ from node y to some node z, for any node $b \in \rho, b \neq y$ we use the notation $\bar{\rho}(b)$ to refer to the set of all clauses derivable from $Peb(\mathcal{G})$ which contain b positively and no node between y and b in ρ negatively. Intuitively, these are all the derivable clauses which can block ρ at b. A **blocking set** $\mathcal{B}$ between a set of nodes S and a node z in $\mathcal{G}$ is a minimal set of clauses in the current resolution configuration which blocks every path from any member of S to z. Note that according to the definition of an unblocked path, S is not a blocking set between S and z.

We first point out a few useful facts about $Peb(\mathcal{G})$. First, $Peb(\mathcal{G})$ is Horn and for every variable x, the only initial clause which contains x positively is $ax(x)$. Therefore, in order for memory to contain any clause containing x positively, $ax(x)$ must first be downloaded. Also, since $Peb(\mathcal{G})$ is Horn, every clause can contain at most one positive literal. Also, since all of $Peb(\mathcal{G})$'s initial clauses contain a positive literal, it is impossible to derive a purely negative clause.

Lemma 11: If there exists an unblocked path ρ from node y to z in configuration $\mathcal{C}[t_1]$, then in order to derive some clause C in $\bar{\rho}(z)$ at time $t_f > t_1$, some clause from $pos(y)$ must be in configuration $\mathcal{C}[t_2]$ for some t_2, $t_1 \leq t_2 < t_f$.

Proof: The proof is by strong induction on the length of the subpath of ρ from y to z.

For the base case, suppose there exists an unblocked path from y to z at time t_1 and we want to derive $\bar{\rho}(z)$ when z is the immediate successor of y. Either some clause from $pos(z)$ is already in memory, or not. If it is, we know that it must contain $\neg y$ because ρ is unblocked. Otherwise, we must download $ax(z)$ which also contains $\neg y$ in order to derive $\bar{\rho}(z)$. In both cases, we must resolve away $\neg y$ in order to derive C. Therefore, we must have a clause from $pos(y)$ in memory.

Now suppose that the statement is true for all nodes b which are up to distance $k-1$ away from y along ρ; we want to prove that the statement holds for node z which is distance k away from y along ρ. Consider the last action, $\mathcal{A}[t_f]$ in the derivation of some clause C in $\bar{\rho}(z)$. This action must be a resolution on some node b which appears before z in ρ (because otherwise an earlier clause would be in $\bar{\rho}(z)$.) Assume that C is derived at action $\mathcal{A}[t_f]$ from clauses C_1 and C_2, where C_1 contains z and $\neg b$, and C_2 contains b. Because C does not contain any literals of nodes upstream of z in ρ, C_2 also does not contain any literals of nodes upstream of b in ρ. Therefore, C_2 is in $\bar{\rho}(b)$. Because C_2 blocks ρ, it must have been derived at some time t_3, between t_1 and t_f Thus by the inductive hypothesis, some clause from $pos(y)$ must be in some configuration $\mathcal{C}[t_2]$, where $t_1 \leq t_2 \leq t_3 \leq t_f$, completing the proof. $\square$

Lemma 12: For every entry point variable y, if $ax(y)$'s assignment variables are clamped, then in order to derive any unit clause from $ax(y)$ without using more than $6n+3$ space, there must be some point between downloading $ax(y)$ and using it for the first time, when the resolution configuration contains only clamped assignment variables of $ax(y)$ and at most one negative literal of $ax(y)$.

Proof: The entry point variable y is either g_0^1 or y is one of x_i', d_i, or $\overline{x_i'}$ for some $i \in [1,n]$.

- Case 1: The entry point variable y is g_0^1. Since g_0^1 is the OR-node z_m, it has 7 axioms (1 for each predecessor p_0^{abc}), each of which has the following form:

$$ax(g_0^1) \quad = \quad \left(\bigvee_{j=1}^{n} (O(g_0^1, x_j) \vee O(g_0^1, d_j) \vee O(g_0^1, \overline{x_j})) \vee p_0^{abc}\right) \to g_0^1$$

Therefore $|ax(g_0^1)| = (3n) + 1 + 1$. Note that the size of the current clamping is $3n$.

- Case 2: The entry point variable y is x_i'. The initial clause $ax(x_i')$ has the following form:

$$ax(x_i') \quad = \quad \left(\bigvee_{j=i+1}^{n} (O(x_i', x_j) \vee O(x_i', d_j) \vee O(x_i', \overline{x_j})) \vee \bigvee_{j=1}^{6i+1} y_j^{x_i'}\right) \to x_i'$$

  Therefore $|ax(x_i')| = (3n-3i) + (6i+1) + 1 = 3n+3i+2$. Note that the size of the current clamping is $3n-3i$.

- Case 3: The entry point variable y is d_i. The initial clause $ax(d_i)$ has the following form:

$$ax(d_i) \quad = \quad \left(\bigvee_{j=i+1}^{n} (O(d_i, x_j) \vee O(d_i, d_j) \vee O(d_i, \overline{x_j})) \vee O(d_i, x_i) \vee \bigvee_{j=1}^{6i-1} y_j^{d_i}\right) \to d_i$$

  Therefore $|ax(d_i)| = (3n-3i) + 1 + (6i-1) + 1 = 3n+3i+1$. Note that the size of the current clamping is $3n-3i+1$.

- Case 4: The entry point variable y is $\overline{x_i'}$. The initial clause $ax(\overline{x_i'})$ has the following form:

$$ax(\overline{x_i'}) \quad = \quad \left(\bigvee_{j=i+1}^{n} (O(\overline{x_i'}, x_j) \vee O(\overline{x_i'}, d_j) \vee O(\overline{x_i'}, \overline{x_j})) \vee O(\overline{x_i'}, x_i) \vee O(\overline{x_i'}, d_i) \vee \bigvee_{j=1}^{6i-3} y_j^{\overline{x_i'}}\right) \to \overline{x_i'}$$

  Therefore $|ax(\overline{x_i'})| = (3n-3i) + 2 + (6i-3) + 1 = 3n+3i$. Note that the size of the current clamping is $3n-3i+2$.

So in all cases, when $ax(y)$ is downloaded our space usage is at least $6n+2$ and we have at most one unit of space left to use. In order to resolve anything away from $ax(y)$ without exceeding our space limit, this space must be filled with some negative literal of $ax(y)$.
$\square$

Corollary 13: Let y be any entry point node and let z_1 and z_2 be any pair of internal nodes such that configuration $\mathcal{C}[t_1]$ satisfies:

1. there is an unblocked path from y to z_2 and from z_2 to z_1;
2. no clause from $pos(z_2)$ is in $\mathcal{C}[t_1]$; and
3. all of the assignment variables of $ax(y)$ are clamped.

Also suppose that the next action, $\mathcal{A}[t_1+1]$, is a download of $ax(z_2)$. Then in order to derive $pu(z_1)$, while keeping all assignment variables of $ax(y)$ clamped (without exceeding $6n+3$ space), some clause in $pos(y)$ must be in $\mathcal{C}[t_1]$.

Proof: Assume $pu(z_1)$ is derived at action $\mathcal{A}[t_f]$, $t_f > t_1$. Since there is an unblocked path from y to z_2 and z_2 to z_1 and there is no clause from $pos(z_2)$ in $\mathcal{C}[t_1]$, then there is an unblocked path from y to z_1 in $\mathcal{C}[t_1]$. Since $pu(z_1) \in \bar{\rho}(z_1)$, by Lemma 11, some clause in $pos(y)$ must be in configuration $\mathcal{C}[t_2]$, for some t_2, $t_1 \leq t_2 < t_f$. If $t_1 = t_2$ then we are done. Otherwise, we must derive some clause C in $pos(y)$ at time t_2, $t_1 < t_2 < t_f$. Therefore, $ax(y)$ must be downloaded at some time t_3, $t_1 < t_3 < t_2$. Now by Lemma 12, at configuration $\mathcal{C}[t_3]$, the positive literal z_1 cannot appear in memory. This is a contradiction because we must still have a clause from $pos(z_1)$ in memory until $pu(z_1)$ is derived. □

DEFINITION 5.5: For each entry point node v we use $\gamma(v)$ to refer to v's position in an ordering which is defined as follows:, $\gamma(g_0^1) = 0$ and for each $i \in [1,n]$: $\gamma(\overline{x_i'}) = 3i-2$, $\gamma(d_i) = 3i-1$, and $\gamma(x_i') = 3i$

Informally, the following lemma states that if there is an unblocked path from y_j (an entry point node in widget j) to z, but all nodes below y_j are blocked, then we must have a blocking set of size $3j$, one for each blocked entry point node below y_j.

Lemma 14: Let y_j be an entry point node of widget j which has b other entry points of widget j after it in γ, and let z be any internal node such that the current configuration $\mathcal{C}[t]$ satisfies the following conditions:

1. there is an unblocked path ρ from y_j to z; and
2. for every entry point node v such that $\gamma(v) < \gamma(y_j)$ there is no unblocked path from v to z.

Then for at least $3j-b$ nodes v which are either not reachable from y_j or are immediate successors of y_j, some clause from $pos(v)$ must be in memory at $\mathcal{C}[t]$.

Proof: The proof hinges on the observation that blocking every path from any entry point node v such that $\gamma(v) < \gamma(y_j)$ to z but not blocking ρ requires a blocking set of size at least $3j-b$. Furthermore, the nodes which comprise the blocking set must occur downstream of any such v and upstream of the first internal node of ρ. If widget j is a universal widget, then we can break the proof into three cases, each with two subcases. Either y_j is $\overline{x_j'}$ and $b=2$, y_j is d_j and $b=1$, or y_j is x_j' and $b=0$. In each of these cases either ρ leaves y_j and traverses up through the left side of widget j or it traverses up through the right side of the widget. We will see that each of these six cases is either impossible, or requires the current configuration to hold some clause from $pos(v)$ for at least $3j-b$ nodes v which are either not downstream of y_j or are immediate successors of y_j.

Case 1: (y_j is $\overline{x_j'}$ and $b=2$):

Right Side: In this case ρ leaves $\overline{x_j'}$ via $\overline{x_j}$. We now show that if ρ leaves $\overline{x_j'}$ via $\overline{x_j}$ and then b_j, then there can be no unblocked paths from entry point nodes in widget $j-1$ which join ρ at some unblocked node of b_j because this would form an unblocked path from them to z. There must therefore be a blocking set $\mathcal{B}$ between any entry point in widget $j-1$ and z which does not block ρ. Any such blocking set must separate any lower entry points from any node on ρ. We argue that this set must have size $3j-2$. A potential blocking set would be all of the predecessors of b_j not in $\{d_j, x_j', \overline{x_j}\}$. Either all of b_j's predecessors that are not in $\{d_j, x_j', \overline{x_j}\}$ are in $\mathcal{B}$ or at least one is not. If they are, then we are done because this set has size $3j-2$. Otherwise, there is some predecessor v of b_j not in $\{d_j, x_j', \overline{x_j}\}$ that is not in $\mathcal{B}$ which is furthest upstream.

Either widget $j-1$ is universal or existential. Suppose it is universal. Then the node v is either g_{j-1}^{3j-2} or g_{j-1}^{3j-3}. Every other member of G_{j-1} is an immediate successor of $\overline{x_{j-1}'}$. If v is one of these, we could reach z from $\overline{x_{j-1}'}$ by crossing the edge from $\overline{x_{j-1}'}$ to v, then to b_j, and from there we could use the nodes of ρ downstream of v to reach z. Since v is one of g_{j-1}^{3j-2} or g_{j-1}^{3j-3} then $\mathcal{B}$ must include every node of $\{g_{j-1}^1, \ldots, g_{j-1}^{3j-4}\}$ since each of these nodes is an immediate successor of an entry point node of widget

$j-1$. Also, either g_{j-1}^{3j-3} or a_{j-1} must be in $\mathcal{B}$, and either g_{j-1}^{3j-2} or x_{j-1} must be in $\mathcal{B}$. So $\mathcal{B}$ has the required size of $3j-2$.

Suppose it is existential. Then the node v must be g_{j-1}^{3j-2} since every other node of G_{j-1} is an immediate successor of d_{j-1}. Therefore $\mathcal{B}$ must include every node of $\{g_{j-1}^{1},\ldots,g_{j-1}^{3j-3}\}$. Also, it is clear that either g_{j-1}^{3j-2} must be in $\mathcal{B}$ or x_{j-1} must be. So $\mathcal{B}$ has the required size of $3j-2$.

Left Side: In this case ρ leaves $\overline{x'_j}$ via some (furthest upstream) node v of $\{g_j^1,\ldots,g_j^{3j-1}\}$. Let $g_j^{l_1}$ be the furthest upstream predecessor of v that is not in $\mathcal{B}$ (or v itself if there is none). So $\mathcal{B}$ must have size at least l_1-1. But $g_{j-1}^{l_1},\ldots g_{j-1}^{3j-2}$ are all predecessors of $g_j^{l_1}$. If widget $j-1$ is universal, then by the same argument as used for the right side $\mathcal{B}$ must include every node of $\{g_{j-1}^{l_1},\ldots,g_{j-1}^{3j-4}\}$ and either g_{j-1}^{3j-3} or a_{j-1} must be in $\mathcal{B}$, and either g_{j-1}^{3j-2} or x_{j-1} must be in $\mathcal{B}$. So $\mathcal{B}$ has the required size of $3j-2$. If widget $j-1$ is existential, then by the same argument as used for the right side $\mathcal{B}$ must include every node of $\{g_{j-1}^{l_1},\ldots,g_{j-1}^{3j-3}\}$ and either g_{j-1}^{3j-2} or x_{j-1} must be in $\mathcal{B}$. So $\mathcal{B}$ has the required size of $3j-2$.

Case 2: (y_j is d_j and $b=1$):

Right Side: In this case ρ leaves d_j via $\{b_j,a_j\}$.

We now show that if ρ leaves d'_j via $\{b_j,a_j\}$ there can be no unblocked paths from $\overline{x'_j}$ which join ρ at some unblocked node of $\{b_j,a_j\}$ because this would form an unblocked path from $\overline{x'_j}$ to z. There must therefore be a blocking set $\mathcal{B}$ between $\overline{x'_j}$ (or any entry point in lower widgets) and z which does not block ρ. Any such blocking set must separate any lower entry points from any node on ρ. Note that ρ must pass through a_j. All $3j-1$ nodes of $G_{j-1}\cup\{b_j\}$ are predecessors of a_j. Either all of these nodes are in $\mathcal{B}$ or at least one is not. If they are all in $\mathcal{B}$, then $\mathcal{B}$ has the size required. Widget $j-1$ is either universal or existential. If it is universal, then every node in $\{g_{j-1}^{1},\ldots,g_{j-1}^{3j-4}\}$ is an immediate successor of an entry point node of widget $j-1$, so each must be in $\mathcal{B}$. Therefore only g_{j-1}^{3j-3}, g_{j-1}^{3j-2}, or b_j might not be in $\mathcal{B}$. As we pointed out for the right side of case 1, either g_{j-1}^{3j-3} or a_{j-1} must be in $\mathcal{B}$, and either g_{j-1}^{3j-2} or x_{j-1} must be in $\mathcal{B}$. If widget $j-1$ is existential, then every node in $\{g_{j-1}^{1},\ldots,g_{j-1}^{3j-3}\}$ is an immediate successor of d_{j-1} so they must all be in $\mathcal{B}$, and it is also easy to see that either g_{j-1}^{3j-2} or x_{j-1} must also be in $\mathcal{B}$. So in either case the size of $\mathcal{B}$ is at least $3j-2$. Finally, either b_j or $\overline{x_j}$ must also be in $\mathcal{B}$, so it has the required size of $3j-1$.

Left Side: This case is impossible since d_i's only successors are on the right side of the widget.

Case 3: (y_j is x'_j and $b=0$):

Right Side: In this case ρ leaves x'_j via $\{b_j,a_j\}$.

This case is impossible since both b_j and a_j are immediate successors of the entry point node d_j, so any unblocked path from x'_j to z would also be an unblocked path from d_j to z.

Left Side: In this case ρ leaves x'_j via x_j.

We now show that if ρ leaves via x_j and then some node(s) of G_j, then there must be a blocking set $\mathcal{B}$ of size at least $3j$ to ensure that there are no unblocked paths from $\overline{x'_j}$ or d_j to z. Observe that ρ cannot use any node of g_j^1 through g_j^{3j-1} because each of those nodes is an immediate successor of $\overline{x'_j}$. Therefore, ρ must leave x_j via g_j^{3j}. But g_j^1 through g_j^{3j-1} are all predecessors of g_j^{3j} and as we noted, they are all successors of

$\overline{x'_j}$. They must all therefore be in $\mathcal{B}$. Therefore, $\mathcal{B}$ must have size at least $3j-1$. Since a_j is an immediate successor of the entry point node d_j, a_j must also be included in $\mathcal{B}$, increasing its size to the desired $3j$.

Suppose, on the other hand that widget j is an existential widget. Then the proof breaks into three cases: either y_j is $\overline{x'_j}$ and $b=2$, y_j is d_j and $b=1$, or y_j is x'_j and $b=0$.

Case 1: (y_j is $\overline{x'_j}$ and $b=2$):

In this case ρ must include some node u of $\{g_j^1, \ldots, g_j^{3i-1}\}$. We can ignore any of u's predecessors in $\{\overline{x_j}, d_j, x_j\}$. Let $g_j^{l_1}$ be u's furthest upstream predecessor not in $\mathcal{B}$ (or u itself if there is none). Then $\mathcal{B}$ contains at least l_1-1 members, namely all of $g_j^{l_1}$'s predecessors in G_j. But $g_j^{l_1}$ also has c_j as a predecessor as well as each node in $\{g_{j-1}^{l_1}, \ldots, g_{j-1}^{3j-3}\}$. Either widget $j-1$ is a universal widget or an existential widget.

Suppose widget $j-1$ is universal. The node $g_{j-1}^{l_1}$ to g_{j-1}^{3j-4} must all be in $\mathcal{B}$ since each is an immediate successor of $\overline{x'_{j-1}}$. Also either g_{j-1}^{3j-3} or x_{j-1} must be in $\mathcal{B}$. This brings the size of $\mathcal{B}$ to at least $3j-3$. We now turn our attention to $g_j^{l_1}$'s other predecessor, c_j. If c_j, e_j, f_j, g_{j-1}^{3j-2}, and a_{j-1} are all not in $\mathcal{B}$ then there is an unblocked path from d_{j-1} to z, crossing from d_{j-1} to a_{j-1}, then to g_{j-1}^{3j-2}, and then to e_j, c_j, $g_{j-1}^{l_1}$, and then to u at which point we can use the node of ρ downstream of u to reach z.

Suppose widget $j-1$ is existential. The node $g_{j-1}^{l_1}$ to g_{j-1}^{3j-3} must all be in $\mathcal{B}$ since each is an immediate successor of d_{j-1}. This brings the size of $\mathcal{B}$ to at least $3j-3$. We now turn our attention to $g_j^{l_1}$'s other predecessor, c_j. If c_j, e_j, f_j, g_{j-1}^{3j-2}, and x_{j-1} are all not in $\mathcal{B}$ then there is an unblocked path from x'_{j-1} to z, crossing from x'_{j-1} to x_{j-1}, then to g_{j-1}^{3j-2}, and then to e_j, c_j, $g_{j-1}^{l_1}$, and then to u at which point we can use the node of ρ downstream of u to reach z.

Case 2: (y_j is d_j and $b=1$):

This case is just like case 1, except that we must also block any path from $\overline{x'_j}$ to z by including $\overline{x_j}$.

Case 3: (y_j is x'_j and $b=0$):

This case is very easy since every node that is an immediate successor of x_j or x'_j is also an immediate successor of d_j except for g_j^{3j+1}. Also, every predecessor of g_j^{3j+1} is an immediate successor of d_j. So in order to block any paths from d_j to z_i without blocking ρ, ρ must use g_j^{3j+1} and all of g_j^{3j+1} s $3j$ predecessors must be in $\mathcal{B}$.

In all cases which can occur, for at least $3j-b$ nodes v which are either not downstream of y_j or are successors of y_j, some clause from $pos(v)$ must be in memory at $\mathcal{C}[t]$. $\square$

Let z_i be an internal node in widget i. Informally, the following lemma states that when we download $ax(z_i)$ (with z_i's assignment variables clamped), then in order to derive something from $ax(z_i)$, we must start in the "full" configuration.

Lemma 15: Let z_i be any internal node of widget i and let z be any internal node such that configuration $\mathcal{C}[t_1]$ satisfies:

1. no clause from $pos(z_i)$ is in $\mathcal{C}[t_1]$;
2. all of the assignment variables of $ax(z_i)$ are clamped; and
3. there is an unblocked path from z_i to z;

Also suppose that the next action, $\mathcal{A}[t_1+1]$, is a download of $ax(z_i)$. Then in order to derive $pu(z)$ at some time t_2 after t_1 (without exceeding $6n+3$ space), there must be $6n+3-|ax(z_i)|-X$ positive unit clauses in memory at t_1, where X is the number of assignment variables of z_i, for nodes which are upstream of z_i, including every entry point v such that there is an unblocked path from v to z_i.

Proof:

Either there is an unblocked path from some entry point node to z_i or not. Suppose not–so all paths from entry point nodes to z_i are blocked. Then there must be some blocking set which separates all entry point nodes upstream of z_i from z_i. We will show that every such blocking $\mathcal{B}$ set has size at least $6n+3-|ax(z_i)|-X$. If all entry point nodes to z_i are blocked, then no entry point node can be a predecessor of z_i. Therefore, if widget i is a universal widget, then the only possibilities for z_i are g_i^{3i} and g_i^{3i+1} and if i is an existential widget, then the only possibility for z_i is g_i^{3i+1}.

Suppose widget i is a universal widget. Case 1) z_i is g_i^{3i}. The node g_i^{3i} has $6n+3-|ax(z_i)|-X-1$ predecessors in G_i which are successors of $\overline{x}_i'$, so they must all be included in $\mathcal{B}$. It also has x_i as a predecessor, which in turn is a successor of x_i'. So therefore, x_i must be in $\mathcal{B}$. So $|\mathcal{B}| = 3i = 6n+3-|ax(z_i)|-X$. Case 2) z_i is g_i^{3i+1}. This is very much like the previous case, except that g_i^{3i+1} also has a_i as a predecessor, and it is a successor of x_i'. So it must also be in $\mathcal{B}$. So $|\mathcal{B}| = 3i+1 = 6n+3-|ax(z_i)|-X$.

Suppose on the other hand that widget i is an existential widget. Then z_i is g_i^{3i+1}. The node g_i^{3i+1} has $6n+3-|ax(z_i)|-X-1$ predecessors in G_i which are successors of d_i, so they must all be included in $\mathcal{B}$. It also has x_i as a predecessor, which in turn is a successor of x_i'. So therefore, x_i must be in $\mathcal{B}$. So $|\mathcal{B}| = 3i+1 = 6n+3-|ax(z_i)|-X$.

Suppose, on the other hand that there is an unblocked path from some entry point node to z_i. Let y_j be the earliest entry point node in γ which has an unblocked path to z_i. Suppose y_j is in widget j, $j \leq i$ and widget j contains b other entry points after y_j in γ. Then clearly, for every entry point node v such that $\gamma(v) < \gamma(y_j)$ there is no unblocked path from v to z_i.

Suppose that z_i has a other entry points of widget i after it in γ, $0 \leq a \leq 2$. Then z_i has $3n-3i+a$ clamped assignment variables (i.e. $X = 3n-3i+a$), and has $3i+1-a$ internal nodes as predecessors in widget i. At the moment when $ax(z_i)$ is downloaded there is therefore at most $3i+1-a$ available space in memory before we exceed $6n+3$.

We can apply Lemma 14 to conclude that the current configuration must hold a set of $3j-b$ clauses whose positive variables are associated with nodes that are either not downstream of y_j or are successors of y_j. (We will see later that this set must be composed of exactly $3j-b$ positive unit clauses for those nodes.) We therefore have at most $3i+1-a-3j+b$ space left when we download $ax(z_i)$. Note that there are exactly $3i-a-3j+b$ entry points upstream of z_i such that $\gamma(v) > \gamma(y_j)$. We will show that for each of these entry points, either some clause from $pos(v)$ or $pos(dual(v))$ must be in the current configuration. This will leave us with at most 1 space left, which we will show must be used by $pu(y_j)$. During the proof we will only count each clause (except for $ax(z_i)$) as using 1 space, and we will still reach our maximum of $6n+3$. We will therefore conclude that each clause (except for $ax(z_i)$) must be a positive unit clause, which will complete the proof.

During this process we follow ρ from z_i down to y_j. We show that given our space bound, it is impossible to keep entry points along the way from connecting to ρ and thereby forming unblocked paths to z_i. The path ρ must pass through each widget k, $i > k > j$. There are two cases depending on whether widget k is a universal or an existential widget.

Suppose widget k is a universal widget. Then ρ can either pass through widget k's right side or straight up. Suppose ρ traverses the right side of widget k. Then ρ must include a_k, which has both d_k and x_k' as predecessors. No clause from $pos(a_k)$ can be in memory since ρ is unblocked, therefore by Corollary 13 a clause of $pos(d_k)$ and a clause of $pos(x_k')$ must be. Also, a_k has b_k as a predecessor, which in turn has x_k as a predecessor, so a clause of either $pos(b_k)$ or $pos(\overline{x}_k)$ must be in memory, or else by Corollary 13 a clause of $pos(\overline{x}_k')$ must be. So in all cases at least three units of space are used in widget k.

Suppose ρ traverses straight up the widget k. Then ρ must include some node of g_k^1 through g_k^{3k-2}. Every node of g_k^1 through g_k^{3k-2} has x_k as a predecessor. So either some clause from $pos(x_k)$ is in $\mathcal{B}$ or we can create a new unblocked path from x_k' to z_i by crossing an edge from x_k to the node which ρ uses in G_k. We can then apply Corollary 13 to conclude that some clause from $pos(x_k')$ must be clamped until after $ax(z_i)$ is downloaded. So either some clause from $pos(x_k)$ is in memory, or some clause from $pos(x_k')$ is. Also, since $\overline{x'_k}$ is a predecessor of every node in g_k^1 through g_k^{3k-2}, by Corollary 13, some clause from $pos(x'_k)$ must also be in memory. Finally, since a_k is also a predecessor of every node in g_k^1 through g_k^{3k-2} and d_k is a predecessor of a_k, it is not hard to see that either some clause from $pos(a_k)$ or some clause from $pos(d_k)$ must also be in memory. So in all cases, if widget k is universal and ρ traverses its left side, then at least three units of space are used in widget k.

Suppose, on the other hand, that widget k is an existential widget. The ρ must include some node of g_k^1 through g_k^{3k-2}. Each one of these nodes has x_k, d_k, and $\overline{x_k}$ as immediate predecessors. So we can therefore apply Corollary 13 to conclude that some clause from $pos(d_k)$ must be clamped until after $ax(z_i)$ is downloaded. Also, if there is no clause of $pos(x_k)$ in memory, then we can apply Corollary 13 to conclude that some clause from $pos(x'_k)$ must be clamped until after $ax(z_i)$ is downloaded. The same is true for $\overline{x_k}$.

Since we only have $3k+1-3j+b-1$ space free, the only way we can block all the paths from d_k to the node of $\{g_k^1,\dots,g_k^{3k-2}\}$ used by ρ is either to have some clause from $pos(a_k)$ in memory which would use 1 extra space. If no clause from $pos(a_k)$ is in memory, there must be an unblocked path from d_k to z_i. We can then apply Corollary 13 to conclude that some clause from $pos(d_k)$ must be clamped until after $ax(z_i)$ is downloaded. So either some clause from $pos(d_k)$ is in memory, or some clause from $pos(a_k)$ is. Finally, every node of $\{g_k^1,\dots,g_k^{3k-2}\}$ has $\overline{x'_k}$ as a predecessor. So we can create a new unblocked path from $\overline{x'_k}$ to z_i by crossing an edge from $\overline{x'_k}$ to the node which ρ uses in G_k. We can now apply Corollary 13 to conclude that some clause from $pos(\overline{x'_k})$ must be clamped until after $ax(z_i)$ is downloaded. □

Lemma 16: Let z be an internal node of $\mathcal{G}$ and let t' and t'' be times such that variable z does not appear in any clause of $\mathcal{C}[t']$ and all of z's assignment variables are clamped during the interval $[t',t'']$. Then in order to derive $pu(z)$ at some time t_2, $t' \le t_2 \le t''$ without exceeding $6n+3$ space, $pu(y)$ must be in memory, for all internal predecessors y of z at time t_1, $t' \le t_1 < t_2$ when $ax(z)$ is downloaded.

Proof: At the moment when $ax(z)$ is downloaded, suppose for the sake of contradiction that for at least one internal predecessor y of z, $pu(y)$ is not in memory. By Lemma 15 we must use $6n+3$ space at $ax(z)$'s download and all the clauses in memory at that time are either $ax(z)$, clamped assignment variables of z, or unit clauses not involving y, so there can be no clause of $pos(y)$ in memory. We must therefore download $ax(y)$ to resolve away the negative literal of y from $ax(z)$. When we download $ax(y)$, if one of y's assignment variables is not clamped, then it will have to be downloaded at sometime when memory still holds a positive clause for a node which is downstream of y. By Lemma 12 this would exceed $6n+3$ space. We therefore know that all of y's assignment variables are clamped when $ax(y)$ is downloaded. We can therefore apply Lemma 15 to conclude that after we've downloaded $ax(y)$, $6n+3$ units of memory must be filled with the union of 1) $ax(y)$, 2) $ax(y)$'s clamped assignment variables, 3) positive unit clauses which are upstream of y. But at this point we must still have a clause in memory which was derived from $ax(z)$ and has size at least 2 since it must at least contain y negatively and z positively. So we exceed $6n+3$ space. All of z's internal predecessors must therefore be in memory when $ax(z)$ is downloaded. □

Lemma 17: If an internal node z of widget i has k predecessors and there are $3n+1-k$ units of space clamped in $[t',t'']$, then $ax(z)$ can be downloaded at most once between t' and t''.

Proof: By Lemma 15, $ax(z)$ cannot be downloaded while a clause from $pos(z)$ is in memory. Therefore, in order to download $ax(z)$ for a second time, every clause of $pos(z)$ must be removed from memory. By frugality, a clause C of $pos(z)$ can only be removed from memory if a sub-clause of C exists in memory or after C has been resolved with a clause of $pos(v)$ for some v downstream of z. This means that once all clauses from $pos(z)$ have been removed from memory, a clause containing a positive literal of a node downstream of z must be in memory. This means that there are at least $3n+1-k+1$ units of space clamped after the first time $ax(z)$ has been downloaded, so trying to download it again would exceed the space bound. □

DEFINITION 5.6: For any assignment $\alpha_i \in A_i$, we define $\mathcal{C}_{\alpha_i}$ as the resolution configuration of $Peb(\mathcal{G})$ consisting of the following clauses for each j, $j \leq n-i$: If $\alpha_i(x_j) = 0$, then $pu(x'_j) \in \mathcal{C}_{\alpha_i}$, $pu(d_j) \in \mathcal{C}_{\alpha_i}$, and either $pu(\overline{x_j}) \in \mathcal{C}_{\alpha_i}$ or $pu(\overline{x'_j}) \in \mathcal{C}_{\alpha_i}$. Otherwise, if $\alpha_i(x_j) = 1$, then $pu(\overline{x'_j}) \in \mathcal{C}_{\alpha_i}$, $pu(a_j) \in \mathcal{C}_{\alpha_i}$, and either $pu(x'_j \in \mathcal{C}_{\alpha_i})$ or $pu(x_j) \in \mathcal{C}_{\alpha_i}$.

To prove Theorem 10, we first observe that by Lemma 16 the derivation of $pu(s)$ requires the derivation of $pu(G_n)$. Therefore, Theorem 10 follows from the following theorem, when $i = n$.

Theorem 18: For every i, $0 \leq i \leq n$, for every $\alpha_i \in A_i$, if there exist times t' and t'' such that $\mathcal{C}_{\alpha_i} \subseteq [t', t'']$, then deriving $pu(G_i)$ at t'' from C_{α_i} at time t', using no more than $6n+3$ space, requires that $\psi\lceil_{\alpha_i}$ is QSAT and requires that there are $\Omega(2^k)$ units of time between t' and t'', where k is the number of universally quantified variables among the inner most i variables of F.

Proof: Now that we have proven the above Lemmas, the proof of our main theorem is very similar to the proof of Theorem 7, and proceeds by induction on i. We will essentially argue that the only $6n+3$ space refutation must follow the upper bound, and hence will imply that the formula is QSAT.

The basis is when $i = 0$. In this case $\mathcal{C}_{\alpha_0}$ has size $3n$ and we must show that deriving $pu(g_0^1)$ without exceeding $6n+3$ space requires at least $\Omega(1)$ time. We show that the derivation requires $\Omega(m)$ time, where m is the number of clauses is F.

We first prove that $pu(z_{j-1})$ must be in memory when we download $ax(z_j)$, for each $1 \leq j \leq m$. With $\mathcal{C}_{\alpha_i}$ clamped in memory, we have $3n+3$ units of space to use to derive $pu(z_m) = pu(g_0^1)$. But each $ax(p_j^{xyz})$ has size exactly $3n+2$. This means that there is one unit of space left over. If this space is not filled with $pu(z_{j-1})$, then the negative literal of z_{j-1} cannot be resolved away until one of the 7 axioms of z_{j-1} is downloaded. But since it is impossible to derive an entirely negative clause, the clause C containing the negative literal of z_{j-1} must be accompanied by a positive literal as well, so it must have size at least 2. Note that at this point we cannot download any $ax(p_{j-1}^{xyz})$ because we would exceed our space bound. But if we download any of the axions of z_{j-1} and resolve them with C, the resolvent will still have size 2, now containing the negative literal of some p_{j-1}^{xyz}, and some clause of size 2 containing p_{j-1}^{xyz} will remain in memory until $ax(p_{j-1}^{xyz})$ is downloaded. We must therefore exceed the space bound.

Now suppose for the sake of contradiction that $Q_n x_n \cdots Q_1 x_1 F\lceil_{\alpha_0}$ is not QSAT, which means that α_0 falsifies F. In that case, there must be some clause C_k of F which is falsified by α_0. In order to derive $pu(z_m)$ we must derive $pu(z_{m-1})$, which in turn requires $pu(z_{m-2})$, etc. Therefore, at some point we must derive $pu(z_k)$ and $pu(z_{k-1})$ must be in memory when we download the axiom of z_k which we will use to derive $pu(z_k)$. But since C_k is falsified by α_0, for each p_k^{xyz}, $\mathcal{C}_{\alpha_0}$ will be missing the positive unit clause of some entry point node. In order to remove its negative literal from $ax(p_k^{xyz})$, the axiom for that entry point will have to be downloaded and used. Clearly, by Lemma 12 this cannot be done within the space bound given the size of the clamped set of clauses $\mathcal{C}_{\alpha_0}$.

Induction Step: We will show that in order to derive $pu(G_i)$ without removing any clauses from $\mathcal{C}_{\alpha_i}$ we must necessarily pass through a number of resolution configurations, each of which can have at most one non-unit clause. This will necessarily involve deriving $\mathcal{C}_{\alpha_{i-1}}$ twice, the first time when $\alpha_{i-1} = \alpha_i \cup \{\overline{x_i}\}$ and the second when $\alpha_{i-1} = \alpha_i \cup \{x_i\}$.

We first observe that by Lemma 15, for every internal variable z or widget i, no variable upstream of z, which is not an assignment variable of z, can be in memory at the time when $ax(z)$ is downloaded without exceeding the space bound of $6n+3$. We will use this observation numerous times in our proof.

Case 1: Q_i is a universal quantifier, so G_i is part of a universal widget.

We will show that in order to derive $pu(G_i)$ we must necessarily pass through a number of partial configurations, including deriving $pu(G_{i-1})$ twice, once while $pu(x'_i)$, $pu(d_i)$, and either $pu(\overline{x_i})$ or $pu(\overline{x'_i})$

(the false configuration) are in memory, and once while $pu(\overline{x_i'})$, $pu(a_i)$, and either $pu(x_i)$ or $pu(x'_{n-i+1})$ (the true configuration) are in memory. We prove this by pointing out 12 important partial configurations $\mathcal{C}[t_0]$ to $\mathcal{C}[t_{11}]$, which occur at times t_0 to t_{11} respectively, where $\mathcal{C}[t_0]$ is $\mathcal{C}_{\alpha_i}$ and $\mathcal{C}[t_{11}]$ contains $pu(G_i)$.

In order to derive $pu(G_i)$ at time t_{11}, we must download the axiom of each member of $G_i \cup \{a_i, b_i, x_i'\}$, since $\mathcal{C}_{\alpha_i}$ does not contain any of their positive literals. In fact, it contains no positive literal for any node in any widget j, $j \leq i$. Therefore, we will need to download the axiom of any positive literal we need from this region of the circuit. Also, by Lemma 17, the axioms of any node in $G_i \cup \{a_i, b_i, x_i'\}$ can be downloaded at most once between t_0 and t_{11}.

Let t_{10} be the time $ax(g_i^{3i+1})$ is downloaded. Clearly, $t_{10} < t_{11}$ and $pos(g_i^{3i+1}) \in [t_{10}, t_{11}]$. Also, by Lemma 16, $pu(\{g_i^1, \ldots, g_i^{3i}\}) \cup \{a_i\} \in \mathcal{C}[t_{10}]$. Let t_9 be the time $ax(g_i^{3i})$ is downloaded. Clearly, $t_9 < t_{10}$ and $pos(g_i^{3i}) \in [t_9, t_{11}]$. Also, by Lemma 16 and Corollary 13, $pu(\{g_i^1, \ldots, g_i^{3i-1}\} \cup \{a_i, (x_i, x_i')\}) \in \mathcal{C}[t_9]$. Let t_8 be the time $ax(g_i^{3i-1})$ is downloaded. Clearly, $t_8 < t_9$ and $pos(g_i^{3i-1}) \in [t_8, t_{11}]$. Also, by Lemma 16 and Corollary 13, $pu(\{g_i^1, \ldots, g_i^{3i-2}\} \cup \{a_i, (x_i, x_i'), \overline{x_i'}\}) \in \mathcal{C}[t_8]$. Let t_7 be the time $ax(g_i^1)$ is downloaded. Clearly, $t_7 < t_9$ and $pos(g_i^1) \in [t_7, t_{11}]$. Also, by Lemma 16 and Corollary 13, $pu(G_{i-1} \cup \{a_i, (x_i, x_i'), \overline{x_i'}\}) \in \mathcal{C}[t_7]$.

Our argument now breaks into two parts. In the first part, we consider what steps are necessary to derive $pu(a_i)$ by t_7. In the second part, we consider what steps are necessary to derive $pu(G_{i-1})$ and $pu(\overline{x_i'})$ by t_7. We will see that $pu(G_{i-1})$ will have to be derived twice. Once in order to derive $pu(a_i)$, and then again immediately before t_7.

Let t_5 be the time $ax(a_i)$ is downloaded. Clearly, $t_5 < t_8$ and $pos(a_i) \in [t_5, t_{11}]$. Also, by Lemma 16 and Corollary 13, $pu(G_{i-1} \cup \{b_i, x_i', d_i\}) \in \mathcal{C}[t_5]$. Let t_4 be the time $ax(b_i)$ is downloaded. Clearly, $t_4 < t_5$ and $pos(b_i) \in [t_4, t_5]$. Also, by Lemma 16 and Corollary 13, $pu(G_{i-1} \cup \{\overline{x_i}, x_i', d_i\}) \in \mathcal{C}[t_4]$.

Let t_1 be the time $ax(x_i')$ is downloaded. By Lemma 12, t_1 must occur when no other positive literal of any widget j, $j \leq i$ is in memory. Furthermore, some clause of $pos(x_i')$ must stay in memory until its successor x_i' is downloaded for the last time. Then $pu(x_i)$ must stay in memory until all of its successors are downloaded for the last time, because we can never derive $pu(x_i)$ once every occurrence of x_i' is removed from memory. Therefore $pos(x_i') \in [t_1, t_5]$ and $pos((x_i, x_i')) \in [t_5, t_9]$.

By Lemma 17, $ax(d_i)$ can only be downloaded once between t_1 and t_{11} since there is always at least 1 unit of memory used between t_1 and t_{11}. Let t_2, $t_2 > t_1$ be the time $ax(d_i)$ is downloaded. By Lemma 12, t_2 must occur when no other positive literal of any widget j, $j \leq i$ is in memory, except for (x_i, x_i'). Therefore $pos(d_i) \in [t_2, t_5]$.

By Lemma 17, $ax(\overline{x_i'})$ can only be downloaded once between t_1 and t_5 since there are always at least 2 unit of memory used between t_2 and t_5. Clearly, $ax(\overline{x_i'})$ must be downloaded within this time frame since it is required to derive $pu(b_i)$ by t_5. Therefore, let t_3, $t_5 > t_3 > t_2$ be the time $ax(\overline{x_i'})$ is downloaded within this interval. By Lemma 12, t_3 must occur when no other positive literal of any widget j, $j \leq i$ is in memory, except for (x_i, x_i') and (d_i, a_i). Furthermore, some clause of $pos(\overline{x_i'})$ must stay in memory until its successor x_i' is downloaded for the last time. Then $pu(x_i)$ must stay in memory until all of its successors are downloaded for the last time, because we cannot derive $pu(x_i)$ during t_1 to t_5 once every occurrence of x_i' is removed from memory. Therefore $pos((\overline{x_i}, \overline{x_i'})) \in [t_3, t_4]$.

So $(\overline{x_i}, \overline{x_i'}), d_i, x_i'$ are clamped from a time when no other positive literals of any widget j, $j \leq i$ are in memory, until at least t_4. But at t_4, $pu(G_{i-1})$ must also be derived. So $(\overline{x_i}, \overline{x_i'}), d_i, x_i'$ must be clamped throughout the derivation of $pu(G_{i-1})$. We can therefore apply the induction hypothesis to conclude that deriving $pu(G_{i-1})$ requires that $\psi\lceil_{\alpha_i \cup \{\overline{x_i}\}}$ is QSAT and requires $\Omega(2^k)$ resolution configurations to occur between t_3 and t_4, where k is the number of universally quantified variables among the inner most $i-1$ variables of ψ.

We now proceed with the second half of the argument in which we investigate the necessities of deriving $pu(G_{i-1})$ and $pu(\overline{x_i'})$ by t_7.

Clearly, $t_7 > t_5$ since $pu(a_i)$ must necessarily be in memory when $ax(g_i^1)$ is downloaded at t_7. But at t_5, no occurrence of the positive literal of $\overline{x_i'}$ can be in memory since its inclusion at t_5 would exceed our space bound. We must therefore download $ax(\overline{x_i'})$ again at some time t_6, after t_5 and $pos(\overline{x_i'}) \in [t_6, t_8]$. By Lemma

12, at t_6 only $pu(a_i)$ and $pu((x_i', x_i))$ can be in memory among all the variables associated with any widget j, $j \leq i$. In particular, no space can be devoted to any literal of any node in G_{i-1} at this time. Since we must derive $pu(G_{i-1})$ by t_7, this necessitates that $pu(G_{i-1})$ must be derived a second time at some point between t_6 and t_7. Since $pos(\overline{x_i'})$, $pos(a_i)$, and $pos((x_i', x_i))$ are all clamped throughout this time, we can apply the induction hypothesis to conclude that deriving $pu(G_{i-1})$ requires that $\psi\lceil_{\alpha_i \cup \{x_i\}}$ is QSAT and requires $\Omega(2^k)$ resolution configurations to occur between t_6 and t_7, where k is the number of universally quantified variables among the inner most $i-1$ variables of ψ.

Case 2: Q_i is an existential quantifier, so G_i is part of an existential widget.

We will show that in order to derive $pu(G_i)$, we must necessarily pass through a number of configurations, including deriving $pu(G_{i-1})$, either with $pu(x_i')$, $pu(d_i)$, and either $pu(\overline{x_i})$ or $pu(\overline{x_i'})$ (the false configuration) in memory, or with $pu(\overline{x_i'})$, $pu(d_i)$, and either $pu(x_i)$ or $pu(x_i')$ (the true configuration) in memory. We prove this by pointing out 12 important partial configurations $\mathcal{C}[t_0]$ to $\mathcal{C}[t_{11}]$, which occur at times t_0 to t_{11} respectively, where $\mathcal{C}[t_0]$ is $\mathcal{C}_{\alpha_i}$ and $\mathcal{C}[t_{11}]$ contains $pu(G_i)$. occurring in the partial configuration must stay in place during a certain interval.

In order to derive $pu(G_i)$ at time t_{11}, we must download the axiom of each member of $G_i \cup \{a_i, b_i, x_i'\}$, since $\mathcal{C}_{\alpha_i}$ does not contain any of their positive literals. In fact, it contains no positive literal for any node in any widget j, $j \leq i$. Therefore, we will need to download the axiom of any positive literal we need from this region of the circuit. Also, by Lemma 17, the axioms of any node in $G_i \cup \{e_i, f_i, x_i'\}$ can be downloaded at most once between t_0 and t_{11}.

Let t_{10} be the time $ax(g_i^{3i+1})$ is downloaded. Clearly, $t_{10} < t_{11}$ and $pos(g_i^{3i+1}) \in [t_{10}, t_{11}]$. Also, by Lemma 16, $pu(\{g_i^1, \ldots, g_i^{3i}\}) \cup \{(x_i, x_i')\} \in \mathcal{C}[t_{10}]$. Let t_9 be the time $ax(g_i^{3i})$ is downloaded. Clearly, $t_9 < t_{10}$ and $pos(g_i^{3i}) \in [t_9, t_{11}]$. Also, by Lemma 16 and Corollary 13, $pu(\{g_i^1, \ldots, g_i^{3i-1}\} \cup \{d_i, (x_i, x_i')\}) \in \mathcal{C}[t_9]$. Let t_8 be the time $ax(g_i^{3i-1})$ is downloaded. Clearly, $t_8 < t_9$ and $pos(g_i^{3i-1}) \in [t_8, t_{11}]$. Also, by Lemma 16 and Corollary 13, $pu(\{g_i^1, \ldots, g_i^{3i-2}\} \cup \{d_i, (x_i, x_i'), (\overline{x_i}, \overline{x_i'})\}) \in \mathcal{C}[t_8]$.

Let t_1 be the time $ax(x_i')$ is downloaded. By Lemma 12, t_1 must occur when no other positive literal of any widget j, $j \leq i$ is in memory. Furthermore, some clause of $pos(x_i')$ must stay in memory until its successor x_i' is downloaded for the last time. Then $pu(x_i)$ must stay in memory until all of its successors are downloaded for the last time, because we can never derive $pu(x_i)$ once every occurrence of x_i' is removed from memory. Therefore $pos((x_i, x_i')) \in [t_1, t_{10}]$.

By Lemma 17, $ax(d_i)$ can only be downloaded once between t_1 and t_{11} since there is always at least 1 unit of memory used between t_1 and t_{11}. Let t_2, $t_2 > t_1$ be the time $ax(d_i)$ is downloaded. By Lemma 12, t_2 must occur when no other positive literal of any widget j, $j \leq i$ is in memory, except for (x_i, x_i'). Therefore $pos(d_i) \in [t_2, t_9]$.

By Lemma 17, $ax(\overline{x_i'})$ can only be downloaded once between t_1 and t_{11} since there are always at least 2 units of memory used between t_2 and t_{11}. By Lemma 12, t_3 must occur when no other positive literal of any widget j, $j \leq i$ is in memory, except for (x_i, x_i') and d_i. Furthermore, some clause of $pos(\overline{x_i'})$ must stay in memory until its successor x_i' is downloaded for the last time. Then $pu(x_i)$ must stay in memory until all of its successors are downloaded for the last time, because we cannot derive $pu(x_i)$ again during t_1 to t_{11} if every occurrence of x_i' is removed from memory. Therefore $(\overline{x_i}, \overline{x_i'}) \in [t_3, t_8]$.

Let t_7 be the time $ax(g_i^{3i-2})$ is downloaded. Clearly, $t_7 < t_8$ and $pos(g_i^{3i-2}) \in [t_7, t_{11}]$. Also, by Lemma 16 and Corollary 13, $pu(\{g_i^1, \ldots, g_i^{3i-3}\} \cup \{d_i, (x_i, x_i'), (\overline{x_i}, \overline{x_i'})\}) \in \mathcal{C}[t_7]$. This leaves exactly one unit of space left in memory at t_7. We will show that this memory must either be filled with $pu(c_i)$ or one of $pu(e_i)$ or $pu(f_i)$. If all three of those clauses are not in memory at t_7 then it is easy to see that there must be unblocked paths at that time from at least two of the three entry points of widget $i-1$ to g_i^{3i-2}, since we can use at most one unit of space to block any paths from the entry points of widget $i-1$ to g_i^{3i-2}. By Corollary 13, clauses containing their variables positively must occur in memory at t_7. Since we have at most 1 space free at t_7, this exceeds our space bound.

If $pu(c_i) \in \mathcal{C}[t_7]$, let t_6 be the last time before t_7 such that $\mathcal{C}[t_6]$ does not contain $pu(c_i)$. In order to derive $pu(c_i)$ at time t_6, either the axiom $(\overline{e_i} \vee c_i)$ or the axiom $(\overline{f_i} \vee c_i)$ must have been downloaded at some time t_5 before t_6.

Suppose $(\overline{e_i} \vee c_i)$ was downloaded at time t_5. Then $ax(e_i)$ must be downloaded at some time t_4 before t_5. By Lemma 17, $ax(e_i)$ can be downloaded at most once between t_0 and t_{11} and by an argument very similar to the proof of Lemma 16 $pu(\{g_{i-1}^1, \ldots, g_{i-1}^{3i-2}\} \cup \{\overline{x_i'}, d_i, (x_i, x_i')\}) \subseteq \mathcal{C}[t_4]$. So $\overline{x_i'}, d_i, (x_i, x_i')$ are clamped from a time when no other positive literals of any widget j, $j \leq i$ are in memory, until at least t_4. But at t_4, $pu(G_{i-1})$ must also be derived. So $\overline{x_i'}, d_i, (x_i, x_i')$ must be clamped throughout the derivation of $pu(G_{i-1})$. We can therefore apply the induction hypothesis to conclude that deriving $pu(G_{i-1})$ requires that $\psi\lceil_{\alpha_i \cup \{x_i\}}$ is QSAT and requires $\Omega(2^k)$ resolution configurations to occur between t_3 and t_4, where k is the number of universally quantified variables among the inner most $i-1$ variables of ψ.

Suppose on the other hand that $(\overline{f_i} \vee c_i)$ was downloaded at time t_5. Then $ax(f_i)$ must be downloaded at some time t_4 before t_5. By Lemma 17, $ax(f_i)$ can be downloaded at most once between t_0 and t_{11} and by an argument very similar to the proof of Lemma 16 $pu(\{g_{i-1}^1, \ldots, g_{i-1}^{3i-2}\} \cup \{(\overline{x_i}, \overline{x_i'}), d_i, x_i'\}) \subseteq \mathcal{C}[t_4]$. So $(\overline{x_i}, \overline{x_i'}), d_i, x_i'$ are clamped from a time when no other positive literals of any widget j, $j \leq i$ are in memory, until at least t_4. But at t_4, $pu(G_{i-1})$ must also be derived. So $(\overline{x_i}, \overline{x_i'}), d_i, x_i'$ must be clamped throughout the derivation of $pu(G_{i-1})$. We can therefore apply the induction hypothesis to conclude that deriving $pu(G_{i-1})$ requires that $\psi\lceil_{\alpha_i \cup \{\overline{x_i}\}}$ is QSAT and requires $\Omega(2^k)$ resolution configurations to occur between t_3 and t_4, where k is the number of universally quantified variables among the inner most $i-1$ variables of ψ.

If $pu(c_i) \notin \mathcal{C}[t_7]$, but $pu(e_i)$ is, then the argument is very similar to the argument above when $(\overline{e_i} \vee c_i)$ is downloaded at t_5. If $pu(f_i)$ is, then the argument is very similar to the one when $(\overline{g_i} \vee c_i)$ is downloaded at t_5.

□

6 Putting it all together

Theorem 1: The resolution space problem is *PSPACE*-complete.

Proof: Every unsatisfiable formula has a space n resolution proof, and thus there is a nondeterministic *PSPACE* algorithm guessing a space n proof. By Savitch's theorem, this implies a deterministic PSPACE algorithm. To show *PSPACE*-hardness, from a QBF formula ψ, we construct the associated CNF formula $Peb(\mathcal{G}_{\mathsf{RES}})$. By Theorem 3 and 6, if ψ is QSAT, then there is a resolution derivation of $pu(G_n)$ from $Peb(\mathcal{G}_{\mathsf{RES}})$ which uses $6n+3$ space. Conversely, by Theorem 10, if there is a resolution derivation of $pu(G_n)$ from $Peb(\mathcal{G}_{\mathsf{RES}})$ using $6n+3$ space, then ψ is QSAT. □

Theorem 2: There exist CNF formulas which have linear size resolution proofs that can be verified in space $k+3$, but whose smallest resolution proofs that can be verified in space k have exponential size.

Proof: Let $\psi = \forall x_n \forall x_{n-1} \ldots \forall x_1 F$ be any totally universally quantified QBF which is QSAT, and let $\mathcal{G}_{\mathsf{RES}}$ be the graph obtained from ψ. Since ψ is QSAT, by Theorem 10, there exists a space $6n+3$ resolution proof of $Peb(\mathcal{G}_{\mathsf{RES}})$, and any space $6n+3$ derivation of $Peb(\mathcal{G}_{\mathsf{RES}})$ requires $\Omega(2^n)$ steps. By Lemmas 5 & 6, there exists a space $6n+6$ derivation of $Peb(\mathcal{G}_{\mathsf{RES}})$ which only requires $O(|Peb(\mathcal{G}_{\mathsf{RES}})|)$ steps. □

7 Open Problems

We have shown that resolution space complexity is PSPACE complete. However, can the space be well-approximated? To what extent do our results hold for other proof systems such as the Polynomial Calculus and Cutting Planes systems?

8 Figures

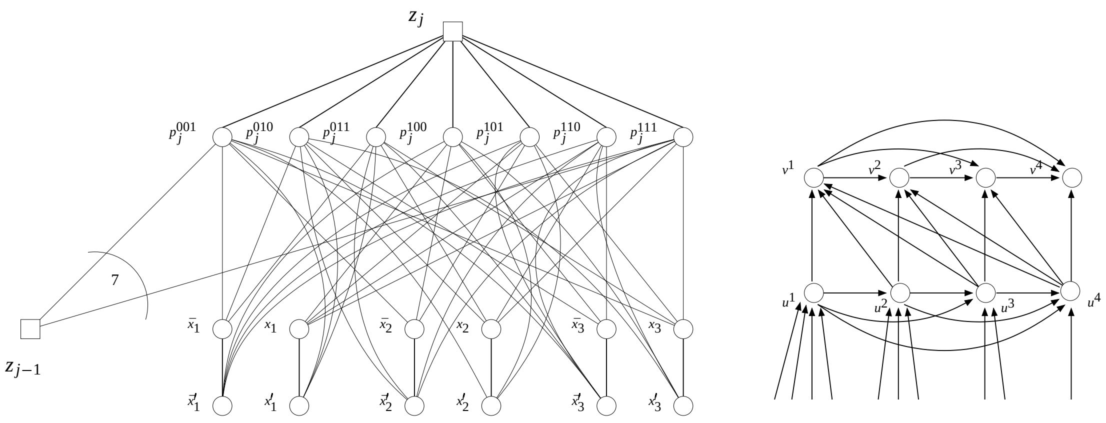


Figure 1: A clause widget for clause $z_j = (x_1 \vee x_2 \vee x_3)$ in $\mathcal{G}$. (All edges are directed from lower nodes toward higher nodes.) And a 4-slide $(\{v^1, v^2, v^3, v^4\}, \{u^1, u^2, u^3, u^4\})$.

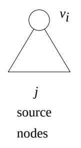
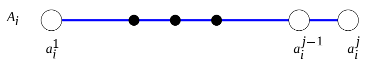
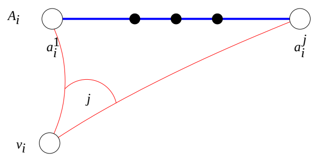
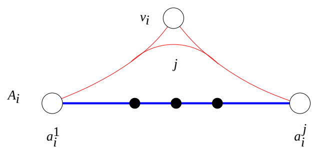
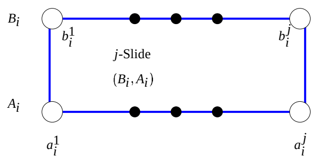
	The node v_i has j source nodes as predecessors.
	For every pair of nodes a_i^j and a_i^k in $A_i = \{a_i^1, \cdots, a_i^m\}$, if $k < j$, then a_i^k is a predecessor of a_i^j.
	The node v_i is a predecessor of all j nodes in A_i.
	The node v_i is a successor of all j nodes in A_i.
 B_i b_i^1 b_i^j j-Slide (B_i, A_i) A_i a_i^1 a_i^j	There is a j-slide from A_i up to B_i.

Figure 2: Legend explaining the components of Figures 3 and 4.

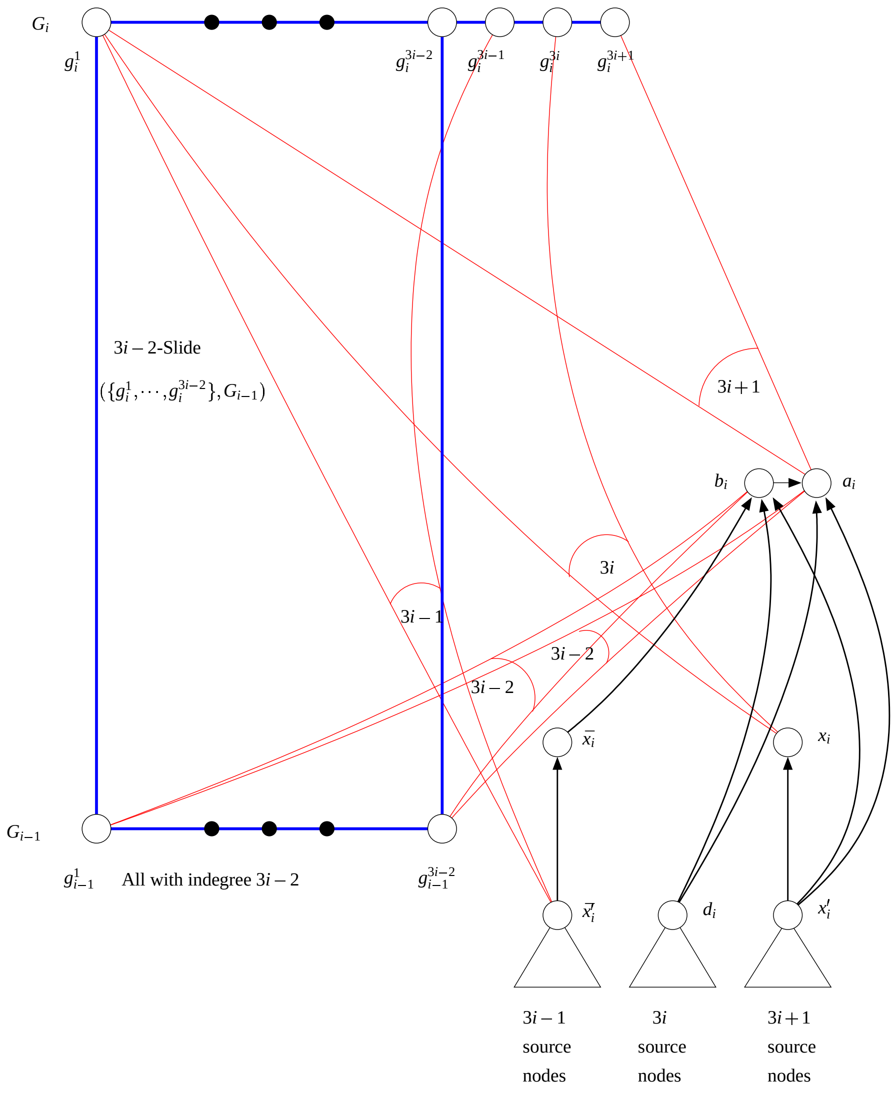


Figure 3: A universal widget.

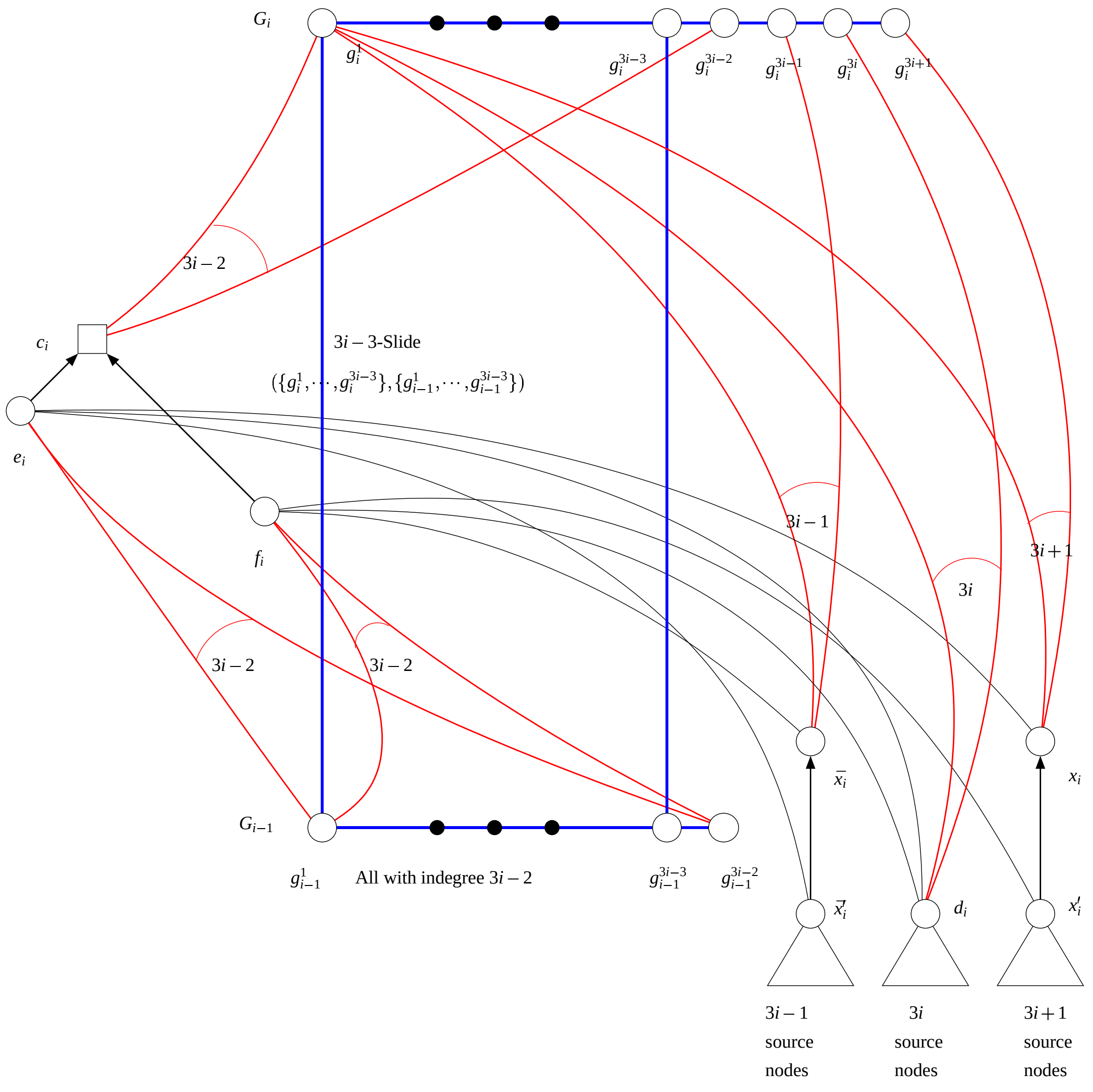


Figure 4: An existential widget.

Acknowledgements

We would like to thank Alex Hertel and Alasdair Urquhart for many very useful conversations, and in particular for suggesting the time/space tradeoff problem and its connection to pebbling.

References

[1] Special Volume on the SAT 2007 Competitions and Evaluations. volume 2, 2006. See also www.satcompetition.org.

[2] M. Alekhnovich, E. Ben-Sasson, A.A. Razborov, and A. Wigderson. Space Complexity in Propositional Calculus. *SIAM Journal of Computing*, Vol. 31, No. 4:1184 – 1211, 2001. Preliminary version: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, 2000, pp. 358-367.

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

[4] Eli Ben-Sasson and Avi Wigderson. Short Proofs Are Narrow – Resolution Made Simple. *Journal of the Association for Computing Machinery*, 48:149–169, 2001. Preliminary version: Proceedings of the 31st Annual ACM Symposium on Theory of Computing, 1999, pp. 517-526.

[5] Stephen Cook. Logic and Mechanical Proof Procedures. 1973. Lectures notes from CSC2409S, Lecture 4, January 17.

[6] A. Darwiche and D. Allen. Optimal time-space tradeoff in probabilistic inference. In *European Workshop on Probabilistic Graphical Models*, 2002. available at www.cs.ucla.edu/~darwiche.

[7] J. Esteban and J. Torán. Space Bounds for Resolution. *Information and Computation*, 171:84 – 97, 2001.

[8] J. Esteban and J. Torán. A Combinatorial Characterization of Treelike Resolution Space. *Electronic Colloquium on Computational Complexity*, 44, 2003.

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

[10] P. Hertel and T. Pitassi. Black White Pebbling is PSPACE Complete. In *Submitted*, 2007.

[11] H. Samulowitz and F. Bacchus. Using sat in qbf. In *"Constraint Programming"*, pages 578–592, 2005.