The Complexity of Propositional Proofs

Alasdair Urquhart*

July 18, 1995

1 Introduction

The classical propositional calculus has an undeserved reputation among logicians as being essentially trivial. I hope to convince the reader that it presents some of the most challenging and intriguing problems in modern logic.

Although the problem of the complexity of propositional proofs is very natural, it has been investigated systematically only since the late 1960s. Interest in the problem arose from two fields connected with computers, automated theorem proving and computational complexity theory. The earliest paper in the subject is a ground-breaking article by Tseitin [59], the published version of a talk given in 1966 at a Leningrad seminar. In the three decades since that talk, substantial progress has been made in determining the relative complexity of proof systems, and in proving strong lower bounds for some restricted proof systems. However, major problems remain to challenge researchers.

The present paper provides a survey of the field, and of some of the techniques that have proved successful in deriving lower bound on the complexity of proofs. A major area only touched upon here is the proof theory of bounded arithmetic and its relation to the complexity of propositional proofs. The reader is referred to the book by Buss [10] for background in bounded arithmetic. The forthcoming book by Krajíček [40] also gives a good introduction to bounded arithmetic, as well as covering most of the basic results in complexity of propositional proofs.

2 Proof systems and simulation

The literature of mathematical logic contains a very wide variety of proof systems. To compare their efficiency, we need a general definition of a proof system. In this section, we give such a definition, together with another that formalizes

*The author gratefully acknowledges the support of the National Sciences and Engineering Research Council of Canada.

the relation holding between two proof systems when one can simulate the other efficiently. The definitions are adapted from Cook and Reckhow [20].

Let Σ be a finite alphabet; we write Σ^* for the set of all finite strings over Σ. The length of a string x is written as $|x|$.

Definition 2.1 *If Σ_1 and Σ_2 are finite alphabets, a function f from Σ_1^* into Σ_2^* is in $\mathcal{L}$ if it can be computed by a deterministic Turing machine in time bounded by a polynomial in the length of the input.*

The class $\mathcal{L}$ of polynomial-time computable functions is a way of making precise the vague notion of "feasibly computable function".

Definition 2.2 *If $L \subseteq \Sigma^*$, a* proof system *for L is a function $f : \Sigma_1^* \to L$ for some alphabet Σ_1, where $f \in \mathcal{L}$ and f is onto. A proof system f is* polynomially bounded *if there is a polynomial $p(n)$ such that for all $y \in L$, there is an $x \in \Sigma_1^*$ such that $y = f(x)$ and $|x| \leq p(|y|)$.*

The crucial property of a proof system as defined above is that, given an alleged proof, there is a feasible method for checking whether or not it really is a proof, and if so, of what it is a proof. A standard axiomatic proof system for the tautologies, for example, can be brought under the definition by associating the following function f with the proof system $\mathcal{F}$: if a string of symbols σ is a legitimate proof in $\mathcal{F}$ of a formula A, then let $f(\sigma) = A$, if it is not a proof in $\mathcal{F}$ then let $f(\sigma) = T$, where T is some standard tautology, say $P \vee \neg P$.

Let us recall here some of the basic definitions in computational complexity theory (for details the reader is referred to [32, 36, 46]). A set of strings is in the class $\mathcal{P}$ ($\mathcal{N}P$) if it is recognized by a deterministic (non-deterministic) Turing machine in time polynomial in the length of the input. A set of strings is in the class $co{-}\mathcal{N}P$ if it is the complement of a language in $\mathcal{N}P$. In more logical terms, a set S of strings is in $\mathcal{P}$ if its characteristic function is in $\mathcal{L}$, while it is in $\mathcal{N}P$ if the condition $y \in S$ can be expressed in the form $(\exists x)(|x| \leq p(|y|) \wedge R(x, y))$, where p is a polynomial, and R is a polynomial-time computable relation. Thus $\mathcal{P}$ is the polynomial-time analogue of the recursive sets, while $\mathcal{N}P$ corresponds to the recursively enumerable sets. Thus the basic question $\mathcal{P} =?\mathcal{N}P$ is the polynomial-time analogue of the halting problem.

The importance of our main question for theoretical computer science lies in the following result of Cook and Reckhow [20].

Theorem 2.1 *$\mathcal{N}P = co{-}\mathcal{N}P$ if and only if there is a polynomially-bounded proof system for the classical tautologies.*

Proof. If $\mathcal{N}P = co{-}\mathcal{N}P$ then since the set TAUT of classical tautologies is in $co - \mathcal{N}P$, TAUT would be in $\mathcal{N}P$, that is to say, there would be a non-deterministic Turing machine M accepting TAUT. Let f be the function such that $f(x) = y$ if and only if x encodes a computation of M that accepts y; then f is a polynomially-bounded proof system for TAUT.

Conversely, let us assume that there is a polynomially-bounded proof system for TAUT. Let L be a language in $\mathcal{N}P$. By the basic $\mathcal{N}P$-completeness result of Cook [16], L is reducible to the complement of TAUT in the sense that there is a function $f \in \mathcal{L}$ so that for any string x, $x \in L$ if and only if $f(x) \notin TAUT$. Hence a nondeterministic polynomial-time procedure for accepting the complement of L is: on input x, compute $f(x)$ and accept x if $f(x)$ has a proof in the proof system. Hence, $\mathcal{N}P$ is closed under complementation, so $\mathcal{N}P = co{-}\mathcal{N}P$. $\square$

This equivalence result underlines the very far-reaching nature of the widely believed conjecture $\mathcal{N}P \neq co{-}\mathcal{N}P$. The conjecture implies that even ZFC, together with any true axioms of infinity that are thought desirable (provided that they have a sufficiently simple syntactic form) is not a polynomially-bounded proof system for the classical tautologies (where we take a proof of $TAUT(\ulcorner A \urcorner)$ as a proof of the tautology A).

We can say nothing of interest about the complexity of such powerful proof systems as the above (in effect, the strongest we can imagine). We can, however, order proof systems in terms of complexity, and prove some non-trivial separation results for systems low down in the hierarchy.

Definition 2.3 *If $f_1 : \Sigma_1^* \to L$ and $f_2 : \Sigma_2^* \to L$ are proof systems for L, then f_2* p-simulates *f_1 provided that there is a polynomial-time computable function $g : \Sigma_1^* \to \Sigma_2^*$ such that $f_2(g(x)) = f_1(x)$ for all x.*

Thus g is a feasible translation function that translates proofs in f_1 into proofs in f_2. We have assumed in the above definition that the language of both proof systems is the same. Reckhow's thesis [50, §5.1.2] contains a more general definition of p-simulation that eliminates this restriction. It is easy to see that the p-simulation relation is reflexive and transitive, and also that the following theorem can be proved from the definitions.

Theorem 2.2 *If a proof system f_2 for L p-simulates a polynomially bounded proof system f_1, then f_2 is also polynomially bounded.*

The symmetric closure of the p-simulation relation is an equivalence relation; thus we can segregate classes of proof systems into equivalence classes within which the systems are "equally efficient up to a polynomial".

3 A map of proof systems

Since the complexity class $\mathcal{P}$ is closed under complementation, it follows that if $\mathcal{P} = \mathcal{N}P$ then $\mathcal{N}P = co - \mathcal{N}P$. This suggests that we might attack the problem $\mathcal{P} =?\mathcal{N}P$ by trying to prove that $\mathcal{N}P \neq co - \mathcal{N}P$; by Theorem 2.1, this is the same as trying to show that there is no polynomially-bounded proof system for the classical tautologies. This line of research was first suggested in papers by Cook and Reckhow [19, 20]. At the moment, the goal of settling the question $\mathcal{NP} \neq co-\mathcal{NP}$ seems rather distant. However, progress has been made in classifying the relative complexity of well known proof systems, and in proving lower bounds for restricted systems. An attractive feature of the research programme is that we can hope to approach the goal step by step, developing ideas and techniques for simpler systems first.

The diagram in Figure 1 is a map showing the relative efficiency of various systems. The boxes in the diagram indicate equivalence classes of proof systems under the symmetric closure of the p-simulation relation. Systems below the dotted line have been shown to be not polynomially bounded, while no such lower bounds are known for those that lie above the line. Hence, the dotted line represents the current frontier of research on the main problem. Although systems below the line are no longer candidates for the role of a polynomially bounded proof system, there are still some interesting open problems concerning the relative complexity of such systems. Questions of this sort, although not directly related to such problems as $\mathcal{NP} =?co-\mathcal{NP}$, have some relevance to the more practical problem of constructing efficient automatic theorem provers. Although the more powerful systems above the dotted line are the current focus of interest in the complex of questions surrounding the $\mathcal{NP} =?co-\mathcal{NP}$ problem, the systems below allow simple and easily mechanized search strategies, and so are still of considerable interest in automated theorem proving.

An arrow from one box to the other in the diagram indicates that any proof system in the first box can p-simulate any system in the second box. In the case of cut-free Gentzen systems, this simulation must be understood as referring to a particular language on which both systems are based. An arrow with a slash through it indicates that no p-simulation is possible between any two systems in the classes in question. If a simulation is possible in the reverse direction, then we can say that systems in one class are strictly more powerful than systems in the other (up to a polynomial). The diagram shows that all such questions of relative strength have been settled for systems below the dotted line, with the exception of the case of the relative complexity of resolution and cut-free Gentzen systems where connectives other than the biconditional and negation are involved.

The diagram shows only a selection from the wide variety of proof systems that have been considered in the literature of logic, automatic theorem proving and combinatorics. A more detailed diagram, showing a wider selection of proof systems, though not reflecting work after 1976, is to be found in Reckhow [50].

Before proceeding to consideration of particular proof systems, let us fix notation. We assume an infinite supply of propositional variables and their negations; a variable or its negation is a *literal*. We say that a variable P and its negation $\sim P$ are *complements* of each other; we write the complement of a literal l as $\bar{l}$. A finite set of literals is a *clause*; it is to be interpreted as the disjunction of the literals contained in it. A clause *mentions* a literal l if either l or $\bar{l}$ is in the clause. The *length* of a clause is the number of literals in it. We shall sometimes write a clause by juxtaposing the literals in it.

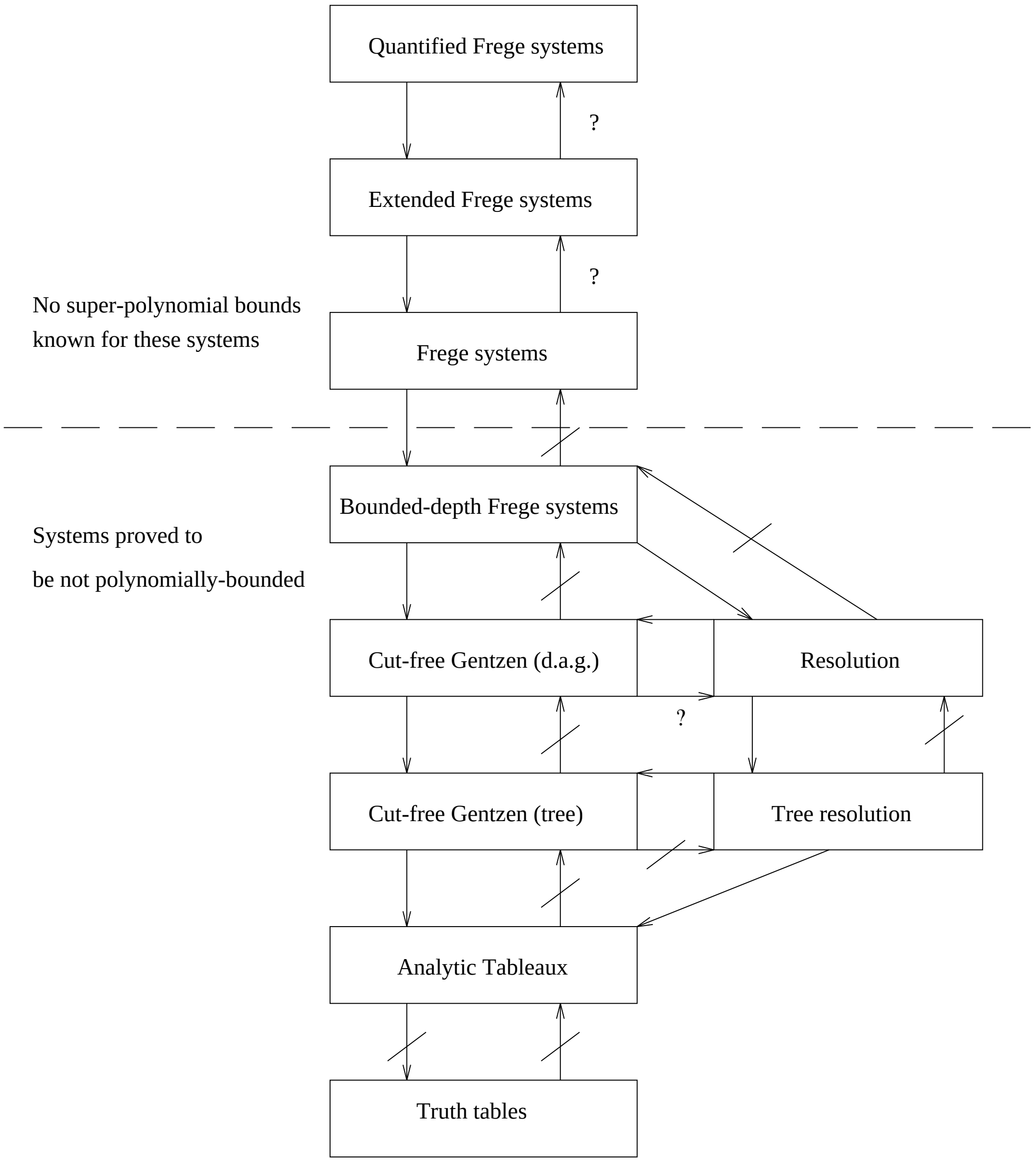

Figure 1: Proof system map

An *assignment* is an assignment of truth-values to a set of propositional variables; some variables may remain unset under an assignment. If Σ is a set of clauses, and ϕ an assignment, then we write $\Sigma\lceil\phi$ for the set of clauses that results from Σ by replacing variables by their values under ϕ and making obvious simplifications. That is to say, if a clause in Σ contains a literal made true by ϕ, then it is removed from the set, while if a literal in a clause is falsified by ϕ then it is removed from the clause. The notation $[l := 1]$ denotes the assignment that sets the literal l to 1 and is otherwise undefined, similarly for $[l := 0]$.

It is useful to fix terminology relating to graphs and trees here. A *graph* consists of a finite set of vertices, a finite set of edges and an incidence relation so that every edge is incident with exactly two distinct vertices (the endpoints of the edge). That is to say, the graphs considered here can contain multiple edges, but not loops; a graph is *simple* if it has at most one edge between any two vertices. Trees should be visualized as genealogical trees, with the root at the top; the nodes immediately below a given node in a tree are its children. The *depth of a tree* T, written $Depth(T)$, is the maximum length of a branch in T.

Derivations in a proof system can be represented either as trees, or as sequences of steps (where a step could be a formula or a sequent). It is normal in the proof-theoretic literature to represent derivations as trees. It is clear, though, that this representation is inefficient, since a step must be repeated every time it is used. If S is a proof system, we denote the corresponding proof system in which derivations are represented as trees by S_{Tree}, reserving the notation S for the system in which derivations are represented as sequences.

4 Analytic Tableaux

The method of analytic tableaux, or truth trees, is employed in many introductory texts; it is given a particularly elegant formulation in Smullyan's monograph [56]. Here we shall only consider the simple form of the method where all formulas are clauses. If Σ is a contradictory set of clauses, then a *tableau for* Σ is a tree in which the interior nodes are associated with clauses from Σ; if a node is associated with a given clause, then the children of that node are labeled with the literals in the clause (the root of the tree remains unlabeled). A tableau for Σ is a *refutation of* Σ if every branch in the tableau is closed (i.e. contains a literal and its negation). We define the *size* of a tableau refutation as the number of interior nodes in the tableau (this measure of complexity, omitting the leaves of the tree, is convenient for inductive proofs). If Σ is a set of clauses, then $t(\Sigma)$ is defined to be the minimum size of a tableau refutation of Σ. Because of the simple structure of tableau refutations, it is possible to prove exact lower bounds on their complexity. The basic tools are the following lemmas.

Lemma 4.1 *In a tableau refutation of minimal size, no branch contains repeated literals.*

Proof. If a tableau refutation contains a branch with repeated literals, then it can be pruned as follows. Let T be a subtree of the tableau whose root is associated with a clause containing a literal l, and this literal l labels a node in the tableau on the path from the root of the tableau to T. Replace T with the immediate subtree of T whose root is labeled with l, but replacing the label on this subtree with the label on the root of T. The resulting tableau is still closed, and is smaller than the original. $\square$

Lemma 4.2 *If Σ is an unsatisfiable set of clauses, then $t(\Sigma)$ satisfies the recursive equation*

$$t(\Sigma) = \min\{t(\Sigma\restriction[l_1 := 1]) + \ldots + t(\Sigma\restriction[l_n := 1]) + 1 : l_1 \vee \ldots \vee l_n \in \Sigma\}.$$

Proof. For $C = l_1 \vee \ldots \vee l_n \in \Sigma$, let T be a tableau refutation of Σ that is minimal among refutations that have C associated with their root. Let $T_1, \ldots, T_n$ be the immediate subtrees of T having $l_1, \ldots, l_n$ as labels on their roots. By Lemma 4.1, the literal l_i does not occur in T_i below the root of T_i; the complement of l_i may occur as the label of at least one leaf in T_i. Thus if we remove from T_i the leaves labeled with $\overline{l_i}$, the result is a refutation of $\Sigma\restriction[l_i := 1]$. Hence the size of T_i is $t(\Sigma\restriction[l_i := 1])$, so that the size of T is

$$t(\Sigma\restriction[l_1 := 1]) + \ldots + t(\Sigma\restriction[l_n := 1]) + 1.$$

Choosing C to minimize this function, we obtain the equation of the lemma. $\square$

A truth table for a formula with n variables, represented as a vector of 0's and 1's, has length 2^n, so that the truth table method is inefficient for large values of n. Of course, we are only considering asymptotic complexity measures here. In practice, the truth table method may be quite efficient for formulas containing a small number of variables, given a reasonably sophisticated implementation. It is easy, however, to find contradictory sets of clauses containing n variables that can be proved quickly by elementary proof methods, for example the sets A_n containing all the variables $P_1, \ldots, P_n$ together with the formula $\sim P_1 \vee \ldots \vee \sim P_n$. The set A_n has a tableau refutation of size $n + 1$.

Somewhat surprisingly, there are cases where truth tables are more efficient than analytic tableaux. This fact was first observed by Marcello D'Agostino, who proved the next result [22].

Theorem 4.1 *The analytic tableau proof system cannot p-simulate the method of truth tables.*

Proof. Let Π_n be the set of all clauses of length n in n variables. For any literals l_1 and l_2 in Π_n, the sets of clauses $\Pi_n\restriction[l_1 := 1]$ and $\Pi_n\restriction[l_2 := 1]$

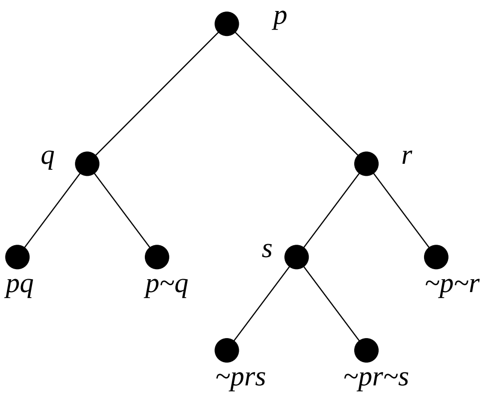

Figure 2: $\Sigma(T) = \{pq, p{\sim}q, {\sim}prs, {\sim}pr{\sim}s, {\sim}p{\sim}r\}$

are logically isomorphic (that is to say, one can be obtained from the other by permuting variables and replacing literals by their complements). Hence, $t(\Pi_n \restriction [l_1 := 1]) = t(\Pi_n \restriction [l_2 := 1])$. It follows by Lemma 4.2 that $t(\Pi_n)$ can be computed by the recursion: $t(\Pi_1) = 2$, $t(\Pi_{n+1}) = (n+1)t(\Pi_n) + 1$. This leads to the explicit formula

$$t(\Pi_n) = n! \left(1 + \frac{1}{1!} + \frac{1}{2!} + \ldots + \frac{1}{n!}\right),$$

asymptotic to $e.n!$. By Stirling's approximation, $(2^n)^c = o(n!)$ for any fixed c, completing the proof. $\square$

Although analytic tableaux work well on simple examples, there are cases where any tableau refutation necessarily contains a great deal of repetition. This is shown by a set of examples due to Cook [17]. Cook's construction associates a set of clauses with a labeled binary tree as follows. Let T be a binary tree in which the interior nodes are labeled with distinct variables. We associate a set of clauses $\Sigma(T)$ with T, in such a way that each branch b in $\Sigma(T)$ has a clause $C_b \in \Sigma(T)$ associated with it. The variables in C_b are those labeling the nodes in b; if P is such a variable, then P is included in C_b if b branches to the left below the node labeled with P, otherwise C_b contains ${\sim}P$. Figure 2 shows a simple example.

Cook's clauses are the sets of clauses $\Sigma_n = \Sigma(T_n)$ associated with the complete binary tree T_n of depth n. To include the case where $n = 0$, we take T_0 to consist of a single node, counted as an interior node; the set of clauses $\Sigma(T_0)$ is $\{\Lambda\}$, where Λ is the empty clause.

If one of the variables in $\Sigma(T)$ is set to 0 or 1, then the resulting simplified set of clauses is also of the form $\Sigma(T')$ for some binary tree T'. Let l be a literal in $\Sigma(T)$, and P the variable in l. Define $T\lceil[l := 1]$ to be the tree resulting from T by replacing the subtree whose root is labeled with P by either its immediate left or right subtree, depending on whether l is negated or not. Then it is easy to see that $\Sigma(T)\lceil[l := 1] = \Sigma(T\lceil[l := 1])$.

The next lemma allows us to compute $t(T) = t(\Sigma(T))$ directly from the structure of T.

Lemma 4.3 *The function $t(T)$ satisfies the following recursion equations:*

1. *If T has only one node, then $t(T) = 1$;*

2. *If T has immediate subtrees U and V, then*

$$t(T) = t(U).t(V) + \min\{t(U), t(V)\}.$$

Proof. If T has only one node, then $\Sigma(T) = \{\Lambda\}$, so $t(T) = 1$ (recall that by our convention, the unique node in a one-node tree counts as an interior node).

Assume the recursion equations hold for trees of size less than that of T, and let T have immediate subtrees U and V. Let $C = l_1 \vee \ldots \vee l_k$ be a clause in $\Sigma(T)$ that is associated with a branch ending in a leaf in U (the argument for branches in V is symmetrical). Define U_j for $2 \leq j \leq k$ to be the labeled tree $U\lceil[l_j := 1]$. Let $t_C(T)$ be the size of a minimal tableau in which C is associated with the root. Then by Lemma 4.2,

$$\begin{aligned} t_C(T) &= t(T\lceil[l_1 := 1]) + \ldots + t(T\lceil[l_k := 1]) + 1 \\ &= t(V) + \sum_{j=2}^{k}[t(V).t(U_j) + \min\{t(V), t(U_j)\}] + 1 \\ &\quad \text{(by the induction hypothesis)} \\ &= t(V)[1 + \sum_{j=2}^{k} t(U_j)] + \sum_{j=2}^{k} \min\{t(V), t(U_j)\} + 1. \qquad (1) \end{aligned}$$

By Lemma 4.2 again,

$$1 + \sum_{j=2}^{k} t(U_j) \geq t(U), \qquad (2)$$

so by (1),

$$\begin{aligned} t_C(T) &\geq t(V)[1 + \sum_{j=2}^{k} t(U_j)] + \min\{t(V), [1 + \sum_{j=2}^{k} t(U_j)]\} \\ &\geq t(V).t(U) + \min\{t(V), t(U)\}. \qquad (3) \end{aligned}$$

For the opposite inequality, assume that $t(U) \leq t(V)$ and that P is the variable labeling the root of T. Let l_1 be P or $\sim P$ according to whether U is the left or right subtree of T, let $l_2 \vee \ldots \vee l_k$ be the clause associated with the root of a minimal tableau refutation of $t(U)$, and C be the clause $l_1 \vee \ldots \vee l_k$. Then for every j, $t(U_j) \leq t(V)$, so that by (1),

$$\begin{aligned} t_C(T) &= t(V)[1 + \sum_{j=2}^{k} t(U_j)] + \sum_{j=2}^{k} t(U_j) + 1 \\ &= t(V).t(U) + t(U), \end{aligned} \tag{4}$$

completing the proof. □

Theorem 4.2 *1. The clauses Σ_n satisfy the recursion equations: $t(\Sigma_0) = 1$, $t(\Sigma_{n+1}) = t(\Sigma_n).[t(\Sigma_n) + 1]$;*

2. The asymptotic behaviour of the function $t(\Sigma_n)$ is given by $t(\Sigma_n) \sim 2^{c2^n}$, where $0.67618 < c < 0.67819$.

Proof. The left and right subtrees of the complete binary tree T_n are isomorphic, so the first claim follows immediately from Lemma 4.3.

Let $z_n = t(\Sigma_n)$; we wish to estimate the growth of z_n. Taking logarithms to the base 2, we have by the first part of the lemma,

$$\log z_{n+1} = 2 \log z_n + \log(1 + 1/z_n), \tag{5}$$

hence

$$\log z_{n+1} = 2^{n-1} + 2^{n-2} \log(1 + 1/z_1) + \ldots + \log(1 + 1/z_n), \tag{6}$$

so that

$$\frac{\log z_{n+1}}{2^{n+1}} = \frac{1}{2} + \frac{\log(1 + 1/z_1)}{4} + \frac{\log(1 + 1/z_2)}{8} + \ldots + \frac{\log(1 + 1/z_n)}{2^{n+1}}. \tag{7}$$

The right hand side of (7) converges rapidly; we find $c = 0.67618634966\ldots$ from $n = 5$. □

The preceding theorem is due to Cook; a version of it appeared without proof in [19]. Cook's original unpublished proof [17] contains a gap; the proof above is joint work of Cook and the present author. Murray and Rosenthal [45] prove a lower bound of $2^{2^{n-1}}$ for the size of analytic tableau refutations of Σ_n.

Theorem 4.2 has some significance for automated theorem proving based on simple tableau methods. The set Σ_6 contains only 64 clauses of length 6, but the minimal tableau refutation for Σ_6 has 10,650,056,950,806 interior nodes. This shows that any practical implementation of the tableau method must incorporate routines to eliminate repetition in tableau construction.

5 Resolution

The resolution rule is a simple form of the familiar cut rule. If Al and $B\overline{l}$ are clauses, then the clause AB may be inferred by the resolution rule, *resolving on* the literal l. A *resolution refutation* of a set of clauses Σ is a derivation of the empty clause from Σ, using the resolution rule. Refutations can be represented as trees or as sequences of clauses; the worst case complexity differs considerably depending on the representation. We shall distinguish between the two by describing the first system as "tree resolution," the second simply as "resolution."

Although resolution operates only on clauses, it can be converted into a general purpose theorem prover for tautologies by employing an efficient method of conversion to conjunctive normal form, first used by Tseitin [59]. Let A be a formula containing various binary connectives such as $\rightarrow$ and $\equiv$; associate a literal with each subformula of A so that the literal associated with a subformula $\sim B$ is the complement of the literal associated with B. If the subformula is a propositional variable, then the associated literal is simply the variable itself. We write l_B for the literal associated with the subformula B. If B is a subformula having the form $C \circ D$, where $\circ$ is a binary connective, then $Cl(B)$ is the set of clauses making up the conjunctive normal form of B. For example, if B has the form $(C \equiv D)$, then $Cl(B)$ is the set of clauses

$$\{\, \overline{l_B}\, \overline{l_C}\, l_D\,,\ \overline{l_B}\, l_C\, \overline{l_D},\ l_B\, \overline{l_C}\, \overline{l_D},\ l_B\, l_C\, l_D \,\}.$$

The set of clauses $Def(A)$ is defined as the union of all $Cl(B)$, where B is a subformula of A.

If A is a tautology, then the set $Def(A) \cup \{\overline{l_A}\}$ is contradictory. Thus we define a proof of A in the resolution system to be a proof of Λ from $Def(A) \cup \{\overline{l_A}\}$. Such a proof of A we shall refer to as a proof by *resolution with limited extension* for the set of connectives (other than $\sim$) occurring in A. In particular, we shall discuss below the system of resolution with limited extension for the biconditional; we refer to this system as $Res(\equiv)$. Note that the set of clauses $Def(A) \cup \{\overline{l_A}\}$ can be derived in time (and space) proportional to the size of A, whereas the same is not true for the conjunctive normal form of $\sim A$ itself (the conjunctive normal form of $P_1 \equiv P_2 \equiv \ldots \equiv P_n$ has size $2^{O(n)}$).

The size of a tree resolution proof is defined as the number of leaves in the tree; if Σ is a contradictory set of clauses, then $tr(\Sigma)$ is defined as the minimal size of a tree resolution refutation of Σ. We shall refer to the clauses at the leaves of a tree resolution derivation as the "input clauses" of the derivation.

Theorem 5.1 *1. Tree resolution p-simulates the method of analytic tableau.*

2. The method of analytic tableau cannot p-simulate tree resolution.

Proof. (1) We shall show that for any inconsistent set of clauses Σ that $tr(\Sigma) \leq t(\Sigma)$. The proof is by induction on the number of variables in Σ. If $\Sigma = \{\Lambda\}$, then $t(\Sigma) = tr(\Sigma) = 1$. Let Σ contain at least one variable, and let $l_1 \vee \ldots \vee l_k$ be the clause associated with the root of a tableau refutation of Σ. Let $T_1, \ldots, T_k$ be the sub-tableaux whose roots are labeled with $l_1, \ldots, l_k$. By Lemma 4.1, none of the clauses associated with the interior nodes of T_i contain the literal l_i, so that the only occurrence of the literal l_i in T_i apart from the root is on leaves labeled $\overline{l_i}$. Hence, by removing these leaves, and the label on the root, the tree T_i becomes a tableau refutation of $\Sigma[l_i := 1]$. By induction hypothesis, $\Sigma[l_i := 1]$ has a tree resolution refutation U_i whose size is less than equal to that of T_i. By adding $\overline{l_i}$ to the appropriate clauses labeling the nodes of U_i, we obtain a tree resolution proof of $\overline{l_i}$ from Σ. Starting from $l_1 \vee \ldots \vee l_k$ and resolving successively on $l_1, \ldots, l_k$, we can combine $U_1, \ldots, U_k$ to obtain a tree resolution refutation of Σ whose size is bounded by the sum of the sizes of $U_1, \ldots, U_k$, completing the induction step.

(2) The examples Σ_n of the preceding section have very small tree resolution refutations. In fact, we can label the interior nodes of T_n with clauses so that it is a tree refutation of Σ_n. □

A sequence of clauses $C_1, \ldots, C_k$ in a resolution derivation is an *irregularity* if each C_i, $i < k$, is a premiss for C_{i+1}, and there is a literal l that appears in C_1 and C_k, but does not appear in any clause C_j, where $1 < j < k$. That is to say, the literal l is removed by resolution from C_1, and is then later re-introduced in a clause depending on C_2. A derivation is *regular* if it contains no irregularity.

Lemma 5.1 *A tree resolution refutation of minimal size is regular.*

Proof. Let $C_1, \ldots, C_k$ be an irregularity in a tree refutation; we shall show how this may be removed while decreasing the size of the refutation. This is accomplished by discarding the first resolution on l, so that for every i, $i < k$, C_i is replaced by a clause D_i, where D_i is a subclause of $C_i l$. If at any point in the new refutation, the literal m resolved upon in the original inference of C_{i+1} from C_i is missing from D_i, then we simply set $D_{i+1} = D_i$.

The resulting refutation is smaller than the original (we have discarded at least one subtree). Since no new irregularities are introduced in the process of removal, the Lemma follows. □

The corresponding lemma for resolution fails. Andreas Goerdt [33] shows that there is an infinite sequence of contradictory sets of clauses having polynomial-size resolution refutations for which the size of any regular resolution refutation grows faster than any fixed polynomial. Goerdt's examples are modified versions of the pigeonhole clauses described in Section 7 below.

Lemma 5.2 *If T is a regular resolution proof of a literal l from a set of clauses Σ, then the result of deleting all occurrences of l from T is a regular resolution refutation of $\Sigma\restriction[l := 0]$.*

Proof. Since T is regular, it can contain no application of resolution where the literal l is resolved on. Hence, $\bar{l}$ cannot occur in any input clause in T, so that the tree resulting from the deletion of l is a refutation of $\Sigma\restriction[l := 0]$. □

Regular tree resolution is closely related to the method of *semantic trees* introduced by Robinson [51] and Kowalski and Hayes [37]. A semantic tree is a binary tree in which the nodes have assignments associated with them. The assignment associated with the root is empty. If ϕ is an assignment associated with an interior node in the tree then the assignments associated with the children of the node are the assignments ϕ_1 and ϕ_2 extending ϕ with $\phi_1(P) = 0$ and $\phi_2(P) = 1$, where P is a variable not in the domain of ϕ. A semantic tree T is a *refutation* of a set of clauses Σ if the variables assigned values in T all belong to Σ and each of the assignments at the leaves of T falsify a clause in Σ.

We can rewrite a regular tree resolution refutation of a set of clauses as a semantic tree by the following technique. First, associate the empty assignment with the root. Second, if $A \vee B$ is a clause in the tree derived by resolution from $A \vee P$ and $B \vee {\sim}P$, and ϕ is associated with the conclusion of the inference, then we associate with the premisses the extensions of ϕ obtained by setting P to 0 and 1 respectively. Conversely, a semantic tree refutation of minimal size can be converted into a resolution refutation by associating with a leaf a clause falsified at that leaf, and then performing resolutions by resolving on the literals labeling the edges.

Regular refutations of a special kind are produced by the *Davis-Putnam procedure*. Given a set of Σ of input clauses, this procedure involves choosing a variable and then forming all possible non-tautologous resolvents from Σ that result from eliminating the chosen variable. This procedure is repeated until the empty clause is produced or no more resolvents can be formed (in which case the input set must be satisfiable). Clearly the refutation produced depends uniquely on the order of elimination adopted. The name of the procedure derives from a well known paper by Davis and Putnam on automated theorem proving [24].

The phrase "Davis-Putnam procedure" is unfortunately ambiguous, since in the literature of automated theorem proving, it refers to a decision procedure for satisfiability involving the recursive construction of a semantic tree. The confusion stems from the fact that during the implementation of the algorithm described in [24], Davis, Logemann and Loveland [23] replaced the original method by this second one, mainly for reasons of space efficiency. In the present article, the phrase "Davis-Putnam procedure" refers to the restricted version of the resolution proof procedure where the refutations are produced by the first method described above.

In the remainder of this section, the lower bounds proved for various forms of resolution are given for the graph-based examples introduced by Tseitin

[59]. This paper of Tseitin is a landmark as the first to give non-trivial lower bounds for propositional proofs; although it pre-dates the first papers on $\mathcal{NP}$-completeness, the distinction between polynomial and exponential growth of proofs is already clear in it.

If G is a graph, then a *labeling* G' of G is an assignment of distinct literals to the edges of G, together with an assignment $\text{Charge}(x) \in \{0, 1\}$ to each of the vertices x in G. If G' is a labeled graph, and x a vertex in G', and $l_1, \ldots, l_k$ the literals labeling the edges attached to x, then $\text{Clauses}(x)$ is the set of clauses equivalent to the conjunctive normal form of the modulo 2 equation $l_1 \oplus \ldots \oplus l_k = \text{Charge}(x)$. That is to say, a clause C in $\text{Clauses}(x)$ contains the literals $l_1, \ldots, l_k$, and the parity of the number of complemented literals in $\{l_1, \ldots, l_k\}$ in C is opposite to that of $\text{Charge}(x)$. The set of clauses $\text{Clauses}(G')$ is the union of all the sets $\text{Clauses}(x)$, for x a vertex in G. Let us write $\text{Charge}(G')$ for the sum modulo 2 of the charges on the vertices of G'; a labeling G' of G is *even* or *odd* depending on whether $\text{Charge}(G')$ is 0 or 1.

Lemma 5.3 *If G is a connected graph, then Clauses(G') is contradictory if and only if the labeling G' is odd.*

Proof. Assume that the labeling G' is odd. If we sum the left-hand side of all the mod 2 equations associated with the vertices of G, the result is 0, because each literal is attached to exactly two vertices, and so appears twice in the sum. On the other hand, the right-hand side sums to 1, by assumption, so the set of equations, and so the set of clauses, is contradictory.

Conversely, suppose $\text{Charge}(G') = 0$. Let x and y be vertices in G connected by an edge e. The set of clauses $\text{Clauses}(G')$ is unchanged if we perform the following operation: replace the literal labeling e by its complement, and replace $\text{Charge}(x)$ and $\text{Charge}(y)$ by their complements. Let us refer to this operation as *transferring a charge* between x and y. If x and y are two vertices in G with $\text{Charge}(x) = \text{Charge}(y) = 1$, then there is a chain of vertices $x = v_1, \ldots, v_j = y$ forming a path from x to y. If we successively transfer a charge from v_1 to v_2 to $\ldots$ v_j, the result is a set of clauses associated with a labeling in which two fewer vertices have an odd charge. Repeating this process, we obtain a labeling in which all vertices have the charge 0. A satisfying assignment is obtained by setting all the literals on the edges to 0. □

For the remainder of this section, we assume that G' is a graph with an odd labeling, and that its edges are labeled with distinct literals; we identify an edge with the literal labeling it. The proof of the preceding lemma shows that any two sets of clauses associated with an odd labeling of a graph G are logically isomorphic, so we shall sometimes write $\text{Clauses}(G)$ to represent any such set.

Let G' be a labeled graph, and l an edge in G'. Define $G' \upharpoonright [l := 0]$ to be the labeled graph resulting from G' by deleting l, and $G' \upharpoonright [l := 1]$ the labeled graph resulting from G' by deleting l and complementing the charges on the vertices incident with l.

Lemma 5.4 *For any graph G' with an odd labeling, Clauses $(G' \upharpoonright [l := 0]) =$ Clauses $(G') \upharpoonright [l := 0]$, and Clauses $(G' \upharpoonright [l := 1]) =$ Clauses $(G') \upharpoonright [l := 1]$.*

Proof. By definition. $\square$

Regular resolution refutations of sets of clauses based on graphs can be visualized in terms of joining together connected subgraphs, as we show in the next two lemmas.

Lemma 5.5 *Let G' be a labeled graph. If T is a resolution proof of a clause C from Clauses(G'), then there is a connected component H' of G' so that T is a resolution proof of C from Clauses(H').*

Proof. By induction on the size of T. $\square$

Let G be an unlabeled connected graph. A *deletion tree for G* is a binary tree labeled with connected subgraphs of G so that the root is labeled with G, and if an interior node is labeled with a subgraph G_1, then the children of that node are labeled with graphs resulting from the deletion of an edge e in G_1. That is, if the deletion of e results in the disconnection of G_1, then the two children are labeled with the two resulting connected subgraphs G_2 and G_3; otherwise, the children are labeled with two copies of G_1 with e deleted.

Lemma 5.6 *Let G be an unlabeled connected graph, and G' an odd labeling of G.*

1. *A deletion tree for G can be labeled with clauses so that it becomes a tree resolution refutation of Clauses(G').*

2. *A tree resolution refutation of Clauses(G') can be labeled with subgraphs of G so that it becomes a deletion tree for G.*

Proof. (1) We prove this by induction on the number of edges in G. If G has no edges, then a deletion tree for G consists of a single node; label this node with Λ. Let T be a deletion tree for a graph G with immediate subtrees T_1 and T_2 whose roots are labeled with graphs G_1 and G_2 obtained by deleting an edge labeled with the literal l in G'. Let G'_1 and G'_2 be the odd labeled components of $G' \upharpoonright [l := 0]$ and $G' \upharpoonright [l := 1]$ respectively. When the deletion of l disconnects G these components must be distinct, and so G_1 and G_2 correspond to the graphs immediately below G in the deletion tree. By induction hypothesis, T_1 and T_2 can be labeled with clauses so that they are regular resolution refutations R_1, R_2 of Clauses(G'_1) and Clauses(G'_2). Since by Lemma 5.4, Clauses $(G'_1) \subseteq$ Clauses $(G') \upharpoonright [l := 0]$, we can add l to the appropriate clauses in R_1 so that it is a regular resolution proof of l from Clauses(G'). Similarly, we can add $\bar{l}$ to R_2 to produce a regular resolution proof of $\bar{l}$ from Clauses(G'). Hence, by labeling the root of T with Λ, we have produced a regular resolution refutation of Clauses(G').

(2) We prove this by induction on the depth of the refutation of Clauses(G'). If the refutation has depth 0, then it consists of Λ alone, so G consists of a single node. Label the root with this node. Let R be a regular tree resolution refutation of Clauses(G') in which the immediate subtrees R_1 and R_2 are proofs of l and $\overline{l}$. By Lemma 5.2 and Lemma 5.4, if we delete l and $\overline{l}$ from R_1 and R_2 respectively, we obtain resolution refutations R_1' and R_2' of Clauses($G' \restriction [l := 0]$) and Clauses($G' \restriction [l := 1]$). By Lemmas 5.3 and 5.5, R_1' and R_2' are refutations of Clauses(G_1') and Clauses(G_2'), where G_1' and G_2' are connected components of $G' \restriction [l := 0]$ and $G' \restriction [l := 1]$ with an odd labeling. By induction hypothesis, the nodes of R_1' and R_2' can be labeled with subgraphs of G_1 and G_2 so that they become deletion trees for G_1 and G_2. If the deletion of l disconnects G, then G_1 and G_2 are distinct components of the resulting disconnected graph. Hence, if we label the root of R with G, the result is a deletion tree for G. □

The above lemma shows that we can compute the complexity function $tr(\Sigma(G))$ directly from the graph G. Thus, we have reduced the problem of proving lower bounds for tree resolution to that of finding graphs that require large deletion trees. The next result is due in its essentials to Tseitin [59].

Theorem 5.2 *Tree resolution cannot p-simulate the Davis-Putnam procedure.*

Proof. For $n > 0$, let the graph G_n consist of $N = 2^n$ vertices $v_1, \ldots, v_N$ with adjacent vertices v_i and v_{i+1} joined by n edges. The set of clauses Clauses(G_n) contains $2^{3n}(1/2 - \mathrm{O}(2^{-n}))$ clauses of size at most $2n$.

Let T be a minimal deletion tree for G_n. Define a branch in T as follows: whenever the children of a node in T are labeled with distinct graphs resulting from the disconnection of the parent graph, then the branch contains the larger of the two sibling graphs. Since it requires the deletion of n edges to disconnect such components, it follows that there are at least $n(n-1)$ interior nodes q in T along the chosen path where the deletion of the chosen edge does not disconnect the graph at the node. At such a node q, the complexity of the subtree rooted at q must be twice that of either of its subtrees. Thus the size of T is at least $2^{n(n-1)}$, showing that $tr(\Sigma(G_n)) = 2^{n(n-1)}$, a function that is not bounded by a polynomial in the size of Clauses(G_n).

On the other hand, there are Davis-Putnam refutations of Clauses(G_n) with sizes linear in the size of the input clauses. The order of elimination is first to eliminate all edges between v_1 and v_2, then between v_2 and v_3 and so on. Since the size of the clauses produced by this procedure is at most $2n$, the result is a refutation whose size is linear in the size of Clauses(G_n). □

By considering a different sequence of graphs, we can find a family of clauses for which the smallest resolution refutations are exponentially big. The basic idea of the lower bound proof given below, from Urquhart [60], is due to Armin Haken [35], who introduced an ingenious "bottle-neck" counting argument to prove the corresponding result for the pigeonhole clauses.

The analysis of refutations of Clauses(G') is facilitated by considering those assignments that make exactly one clause in Clauses(G') false. These are easy to describe. If ϕ is an assignment to the edges of G, and x a vertex in G, then Charge(ϕ, x) is defined to be the sum modulo 2 of the values assigned by ϕ to the edges attached to x. An assignment to the edges of G is *x-critical* if Charge(ϕ, y) = Charge(y) for all vertices y in G except for x. An x-critical assignment falsifies exactly one clause in Clauses(x), while all other clauses in Clauses(G') are satisfied; it is easy to see that this property characterizes x-critical assignments for G.

If G is a connected graph, we say that an assignment ϕ of truth-values to some of the edges of G is *non-separating* if the graph that results by deleting the edges assigned a value by ϕ is connected.

Lemma 5.7 *If ϕ is a non-separating assignment to some of the edges of a connected graph G with an odd labeling, and x is any vertex of G, then ϕ may be extended to an x-critical assignment for G.*

Proof. Fix a spanning tree for G that does not contain any edge assigned a value by ϕ. Assign values arbitarily to any edge not in the spanning tree that has not yet been assigned a value. Proceeding from the leaves of the tree inward towards x, assign values to the edges attached to vertices other than x so that Clauses(y) is satisfied. The resulting assignment must be x-critical since Clauses(G) is contradictory. □

The graphs used in the lower bound for resolution are the expander graphs used by Galil [31] to prove an exponential lower bound for regular resolution, with a small modification to simplify the proof. The expander graph H_m is a simple bipartite graph in which each vertex has degree at most 5 and each side contains m^2 vertices (for brevity we write $n = m^2$). The particular family of expander graphs used here was first defined by Margulis [44]. The exact definition of the graphs is not needed; for the lower bound all that is needed is the expanding property proved by Margulis and stated in the next lemma.

Lemma 5.8 *There is a constant $d > 0$ such that if V_1 is contained in one side of H_m, $|V_1| \leq n/2$, and V_2 consists of all the vertices in the other side of H_m that are connected to vertices of V_1 by an edge, then $|V_2| \geq (1 + d)|V_1|$.*

Proof. See Gabber and Galil [30], who also provide a numerical lower bound for the expansion factor d. □

The graph G_m is obtained from H_m by the following modification. We add $n - 1$ edges to each side of the graph so that each side forms a connected chain. We call the new edges *side edges* and the edges in the original graph *middle edges*. The resulting graph still satisfies Lemma 5.8, and each vertex in it has degree at most 7. Let Ω_m be Clauses(G_m); Ω_m contains at most $128n$ clauses of length at most 7, so the entire set of clauses has size $O(n)$.

We now specify for each m a set of *restrictions* $\mathcal{R}_m$, a family of assignments to some of the edges in G_m. Let d be the constant in Lemma 5.8, and let f be $d/16$. A restriction in $\mathcal{R}_m$ is specified by choosing a set of $\lceil fn \rceil$ middle edges, and then assigning truth-values arbitrarily to the chosen edges. For $P \in \mathcal{R}_m$, we write $E(P)$ for the set of chosen edges in P. Every restriction P in $\mathcal{R}_m$ is non-separating because at least one middle edge must be unset by P.

If C is a clause, and $P \in \mathcal{R}_m$, then we define $Crit(C, P)$ as the set of vertices $x \in G_m$ such that there is an x-critical assignment ϕ extending P so that $\phi(C) = 0$. In a resolution refutation of Ω_m, a clause C is *P-complex*, where $P \in \mathcal{R}_m$, if C is the first clause in the refutation for which $|Crit(C, P)| \geq n/4$; a clause is *complex* if it is P-complex for some P. For every $P \in \mathcal{R}_m$, a P-complex clause must exist in a refutation of Ω_n, because by Lemma 5.7, $|Crit(\Lambda, P)| = 2n$. The complex clauses in a refutation form a "bottle-neck" in that a given clause can only be P-complex for an exponentially small fraction of all $P \in \mathcal{R}_m$.

Lemma 5.9 *If C is a P-complex clause, then at least $\lfloor fn \rfloor$ middle edges are mentioned in C.*

Proof. If C is an input clause in Clauses(x), then $Crit(C, P) = \{x\}$, so we can assume that C is inferred from earlier clauses D and E by the resolution rule. Since the resolution rule is sound, $Crit(C, P) \subseteq Crit(D, P) \cup Crit(E, P)$. Because both $|Crit(D, P)|$ and $|Crit(E, P)|$ are less than $n/4$, $|Crit(C, P)| < n/2$. Let $Crit(C, P) = W_1 \cup W_2$, where W_1 and W_2 are contained in opposite sides of G_m, and $|W_1| \geq |W_2|$; let Y_2 be the set of vertices not in W_2 that are connected to W_1 by a middle edge. Since $|Cover(C, P)| \geq n/4$, $|W_1| \geq n/8$, so by Lemma 5.8, $|Y_2| \geq dn/8$.

Let e be an edge connecting a vertex x in W_1 with a vertex y in Y_2 that is not one of the chosen edges in the restriction P. By the definition of a restriction, there are at least $\lfloor fn \rfloor$ such edges. Since $x \in Crit(C, P)$, there is an x-critical assignment ϕ extending P so that $\phi(C) = 0$. If e is not mentioned in C, then the assignment ϕ' obtained from ϕ by reversing the truth-value of e also falsifies C. The assignment ϕ' is y-critical, and since e is not a chosen edge in P, ϕ' also extends P. This contradicts the assumption that y is not in W_2. $\square$

Theorem 5.3 *There is a constant $c > 1$ such that for sufficiently large m any resolution refutation of Ω_m contains c^n distinct clauses.*

Proof. Let us fix a resolution refutation of Ω_m, with respect to which the notion of complex clause is defined. Every $P \in \mathcal{R}_m$ is associated with exactly one P-complex clause in the refutation, but a given complex clause may have a number of restrictions associated with it. We show that the number of restrictions associated with a given complex clause is exponentially small, so that there must be exponentially many complex clauses in the refutation.

Let C be a complex clause. By Lemma 5.9, there is a set $E(C)$ of middle edges mentioned in C, where $|E(C)| = \lfloor fn \rfloor$. The fraction of restrictions P with

$|E(P) \cap E(C)| = i$ with respect to which C is P-complex is at most 2^{-i}, since the edges in P are set independently. Hence, the ratio between the number of restrictions associated with C and the total number of restrictions is bounded by

$$\sum_{i=0}^{s} \binom{M}{i} \binom{N-M}{s-i} \binom{N}{s}^{-1} 2^{-i}, \tag{8}$$

where $M = |E(C)| = \lfloor fn \rfloor$, $s = |E(P)| = \lceil fn \rceil$, and N is the number of middle edges in H_m.

We can find a bound for (8) by adapting Chvátal's elegant bound [14] for the tail of the hypergeometric distribution. First, we establish an inequality:

$$\begin{aligned}
&\sum_{i=j}^{s} \binom{M}{i} \binom{N-M}{s-i} \binom{i}{j} \binom{N}{s}^{-1} \\
&\quad = \binom{M}{j} \sum_{i=j}^{s} \binom{M-j}{i-j} \binom{N-M}{s-i} \binom{N}{s}^{-1} \\
&\quad = \binom{M}{j} \binom{N-j}{s-j} \binom{N}{s}^{-1} \\
&\quad = \binom{M}{j} \binom{s}{j} \binom{N}{j}^{-1} \\
&\quad \leq \binom{s}{j} \left(\frac{M}{N}\right)^{j}.
\end{aligned} \tag{9}$$

The bound on the ratio (8) follows from this inequality by the computation:

$$\begin{aligned}
&\sum_{i=0}^{s} \binom{M}{i} \binom{N-M}{s-i} \binom{N}{s}^{-1} 2^{-i} \\
&\quad = \sum_{i=0}^{s} \binom{M}{i} \binom{N-M}{s-i} \binom{N}{s}^{-1} \sum_{j=0}^{i} \binom{i}{j} \left(-\frac{1}{2}\right)^{j} \\
&\quad = \sum_{j=0}^{s} \sum_{i=j}^{s} \binom{M}{i} \binom{N-M}{s-i} \binom{i}{j} \binom{N}{s}^{-1} \left(-\frac{1}{2}\right)^{j} \\
&\quad \leq \sum_{j=0}^{s} \binom{s}{j} \left(\frac{M}{N}\right)^{j} \left(-\frac{1}{2}\right)^{j} \\
&\quad = \sum_{j=0}^{s} \binom{s}{j} \left(-\frac{M}{2N}\right)^{j} \\
&\quad = \left(1 - \frac{M}{2N}\right)^{s}.
\end{aligned} \tag{10}$$

Since $N \leq 5n$, $M/2N \geq f/11$, for sufficiently large m. It follows that there must be at least c^n complex clauses in the refutation, where $c = (1 - f/11)^{-f}$.
□

The basic properties of the graph-based clauses that are exploited in the lower bound argument are an "expansion" property (reflected in Lemma 5.9), and an "independence" property (reflected in the fact that in a restriction the chosen middle edges can be set independently). Chvátal and Szemerédi, in a far reaching generalization of the preceding theorem proved a lower bound for sets of randomly chosen clauses, by showing that sets of random clauses satisfy appropriately generalized forms of these properties. Define the *random family of m clauses of length k over n variables* to consist of a random sample of size m chosen with replacement from the set of all clauses of length k with variables chosen from a set of n variables. Chvátal and Szemerédi [15] prove the following result.

Theorem 5.4 *For every choice of positive integers n, c, k such that $k \geq 3$ and $c2^{-k} \geq 0.7$, there is a positive number ϵ such that, with probability tending to one as n tends to infinity, the random family of cn clauses of size k over n variables is unsatisfiable and its resolution complexity is at least $(1+\epsilon)^n$.*

6 Cut-free Gentzen systems

Cut-free Gentzen systems have proved popular in work on automated deduction, since they allow simple search strategies in constructing derivations. In the present section, we consider a sequent calculus G based on the biconditional as the only connective. The axioms of G are sequents of the form $A \vdash A$. The rules of inference of G are as follows:

$$\frac{\Gamma \vdash \Delta}{\Gamma, \Theta \vdash \Delta, \Xi} \ (Thinning)$$

$$\frac{\Gamma, A, B \vdash \Delta \qquad \Gamma \vdash \Delta, A, B}{\Gamma, (A \equiv B) \vdash \Delta} \ (\equiv \vdash)$$

$$\frac{\Gamma, A \vdash \Delta, B \qquad \Gamma, B \vdash \Delta, A}{\Gamma \vdash \Delta, (A \equiv B)} \ (\vdash \equiv)$$

An alternative formulation of G is possible in which the axioms are sequents of the form $\Gamma, A \vdash A, \Delta$, and the thinning rule is omitted. We denote this alternative formulation by G'. It is the version adopted by Smullyan [56, pp. 105-106], and is usually employed in automatic theorem provers; the system of Wang [63] is of this type. The Leningrad group headed by Shanin in their work on computer search for natural logical proofs [52] used a formulation of the second type for the proof search, but then transformed the resulting derivations into simplified derivations in a system of the first type by a pruning procedure.

It is natural to use a system of the second type in a computer search, because if the usual 'bottom-up' search procedure is employed, the thinning rule can (when employed in reverse) result in potentially useful information being discarded. However, as we show below, the two formulations are quite distinct from the point of view of worst case complexity. There are certain sequents for which short proofs can be found only by employing the thinning rule.

Derivations in G have the *subformula property*, that is, any formula occurring in the derivation must occur as a subformula in the conclusion of the derivation. In fact, an analysis of derivations in G shows that *occurrences* of formulas in the derivation can be identified with *occurrences* of formulas in the conclusion. This can be seen by tracing occurrences of formulas step by step up the derivation from the conclusion. Thus in an application of $(\vdash\equiv)$, for example, the occurrences of A in the premisses are to be identified with the occurrence of A in the conclusion of the inference. This identification of occurrences will be used subsequently to prove lower bounds for the proof systems.

The Cut rule

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma \vdash \Delta, A}{\Gamma \vdash \Delta}\ (Cut)$$

is not necessary for completeness, but in some cases results in much shorter derivations. The formula A in the Cut rule is said to be the *cut formula.* The subformula property fails for derivations in the system $G + Cut$ that results by adding the Cut rule to G. However, the property is preserved if we restrict the Cut rule appropriately. We shall say that a derivation of a sequent $\Gamma \vdash \Delta$ is a derivation in G *with the analytic Cut rule* if the derivation belongs to $G + Cut$, and all cut formulas are subformulas of formulas in the conclusion $\Gamma \vdash \Delta$.

We shall now prove some results from [61] that settle the relative complexity of resolution and cut-free Gentzen systems, at least for the case of tautologies involving the biconditional. As in the case of tree resolution, we define the complexity of a derivation in G_{Tree} to be the number of leaves in the derivation (that is, the number of occurrences of axioms). The simulation in the following theorem is due to Tseitin [59].

Theorem 6.1 *The system* $Res(\equiv)_{Tree}$ *p-simulates* G_{Tree}.

Proof: If D is a derivation in G_{Tree} of a sequent $\vdash A$, and $\Theta \vdash \Xi$ is a sequent in D, then any formula B that occurs as an antecedent or consequent formula in $\Theta \vdash \Xi$ is a subformula of A, and hence is assigned a literal l_B as part of $Def(A)$. We construct a clause corresponding to the sequent $\Theta \vdash \Xi$ by forming the disjunction of all the propositional variables $\overline{l_B}$ if B is an antecedent formula, together with the l_B if B is a consequent formula.

The resulting tree of clauses is not a resolution proof, but is easily converted into a resolution derivation of Λ from $Def(A) \cup \{\overline{l_A}\}$ by inserting some added resolvents. In every case except when a sequent is an axiom, we show that we can derive a subclause of the clause corresponding to the sequent. The rules $(\equiv \vdash)$ and $(\vdash \equiv)$ are simulated by forming two resolvents by resolving the clauses corresponding to the premisses against clauses in $Def(A)$, and then resolving these in turn to derive a subclause of the clause corresponding to the conclusion of the inference. The result is a resolution refutation having complexity $O(n)$, where n is the complexity of D. $\Box$

We now define the sequence of biconditional tautologies that form the basis of the lower bounds in this section. For any $n > 0$, let U_n be the formula

$$P_n \equiv P_{n-1} \equiv \ldots \equiv P_1 \equiv P_n \equiv P_{n-1} \equiv \ldots \equiv P_1$$

where we are omitting parentheses according to the convention of association to the right; for example, $A \equiv B \equiv C$ abbreviates $(A \equiv (B \equiv C))$. All the variables in U_n occur exactly twice, so that U_n is a tautology. To distinguish between two occurrences of the same variable P_k, we shall write the first occurrence as P_k^1, the second occurrence as P_k^2. The subformula of U_n beginning with the subformula occurrence P_k^i will be denoted by U_k^i. Thus U_k^2 contains k occurrences of variables, while U_k^1 contains $n + k$ occurrences; in particular, $U_n^1 = U_n$.

If $\Gamma \vdash \Delta$ is a sequent, we use the term *O-assignment* to refer to an assignment of truth-values $\{0, 1\}$ to the occurrences of the variables in $\Gamma \vdash \Delta$. An O-assignment is extended to all the occurrences of subformulas in the sequent by the usual truth table method. The entire sequent takes the value 0 under an O-assignment if all occurrences of formulas in Γ take the value 1, and all the occurrences of formulas in Δ the value 0. It is essential to the notion of O-assignment that distinct truth values can be assigned to different occurrences of the same variable. In particular, by choosing an appropriate O-assignment, it is possible to falsify a tautological sequent. If D is a cut-free derivation of a sequent $\Gamma \vdash \Delta$, then any O-assignment for $\Gamma \vdash \Delta$ can be extended to all the sequents in D; this is possible because of the identification noted earlier between occurrences of formulas in the conclusion and occurrences of formulas in D. The notion of O-assignment was used earlier in a somewhat different form in [60] to prove an exponential lower bound for cut-free Gentzen systems. An exponential lower bound for the tree version of a cut-free Gentzen system was proved earlier by Statman [58].

The formula U_n has 2^{2n} O-assignments associated with it. We are interested only in certain of these. We shall call an O-assignment to U_n *critical* if there is exactly one variable in U_n whose occurrences in U_n are assigned different values. If this variable is P_k, then we say that the O-assignment in question is *k-critical*. All critical O-assignments falsify the formula U_n; a critical O-assignment is uniquely determined by k, and the values the O-assignment gives to the occurrences P_i^1, for $1 \leq i \leq n$, so that there are $n \cdot 2^n$ distinct critical O-assignments for the sequent U_n.

Theorem 6.2 *The minimal complexity of a derivation of* U_n *in the system* G_{Tree} *is* $n \cdot 2^n$.

Proof: Any critical O-assignment for $\vdash U_n$ falsifies the conclusion of the derivation, $\vdash U_n$, and if it falsifies the conclusion of an inference, then it also falsifies one of the premisses. Hence, if ϕ is any critical O-assignment for $\vdash U_n$, we can trace a branch in the derivation from the conclusion to an axiom, so that all the sequents in the branch are falsified by ϕ. If ϕ is a k-critical O-assignment, then the only subformula of U_n whose occurrences have distinct values assigned to them under ϕ is P_k. Thus the axiom at the tip of the branch must have the form $P_k \vdash P_k$, where the antecedent and consequent occurrences of P_k correspond to distinct occurrences of P_k in U_n. It follows from this that if ϕ and ψ are respectively k-critical and j-critical for $j \neq k$, that the branches for ϕ and ψ are distinct. Furthermore, since the axiom at the tip of the branch for ϕ must contain P_k^2, it follows that the branch must contain occurrences of all P_i^1, for $1 \leq i \leq n$. Since distinct k-critical O-assignments give distinct sequences of values to the occurrences P_i^1, all these branches must also be distinct. There are $n \cdot 2^n$ O-assignments for $\vdash U_n$, so that the complexity of the derivation is at least $n \cdot 2^n$. It is easily verified that there is a derivation of this complexity. □

We now show that there are relatively short proofs of U_n in the resolution system. To describe the proofs, it is useful to give a graphical representation of these tautologies. The sets of clauses derived from the sequence of formulas U_n will be represented in the form Clauses(G_n), for a sequence of graphs G_n.

The graph G_n associated with the formula U_n is a planar graph that we describe by giving the co-ordinates of its nodes. G_n has as its nodes the set of points $\{(i,1) : 0 \leq i \leq n\} \cup \{(i,0) : 1 \leq i \leq n-1\}$. The following nodes are joined in G_n: $(i,1)$ to $(i+1,1)$, $(i,0)$ to both $(i,1)$ and $(i+1,0)$, $(n,1)$ to both $(n-1,0)$ and $(1,0)$. The graph may be described as a ladder with a few extra attachments. The labels attached to G_n are as follows. The vertical lines, and the line joining $(n,1)$ to $(n-1,0)$ are labeled with the variables P_n to P_1 from left to right. The horizontal lines joining the points with y co-ordinate 1 are labeled with the variables Q_n^1 to Q_1^1 from left to right; the horizontal lines joining points with y co-ordinate 0 are labeled with the variables Q_{n-1}^2 to Q_2^2 from left to right. The line joining $(n,1)$ to $(1,0)$ is labeled with the variable Q_n^2. The node $(0,1)$ is labeled with 0; all other nodes are labeled 1. The accompanying figure shows the labeled graph corresponding to U_5. A node is shown filled in only if it is labeled with 0.

The set of clauses $Def(U_n) \cup \{\sim Q_n^1\}$, where the variable Q_k^i is correlated with the subformula U_k^i, is identical with Clauses(G_n). The graphs G_n are similar to examples used by Galil ([31, Fig. 3.2.4]) to show that the Davis-Putnam

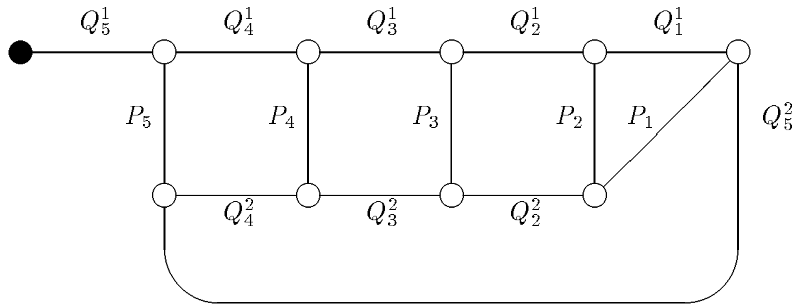

Figure 3: The labeled graph G_5

procedure is very sensitive to the order of elimination adopted in forming resolvents; with one order of elimination, ladder-like graphs result in exponential-size refutations, while a different order gives rise to linear-size refutations.

Theorem 6.3 *The tautologies U_n have proofs in $Res(\equiv)_{Tree}$ of complexity $O(n^2)$.*

Proof: Because the set of clauses $Def(U_n) \cup \{\sim Q_n^1\}$ can be described in terms of the graph G_n, Lemma 5.6 shows that it is sufficient to find a deletion process for G_n in which the underlying tree has $O(n^2)$ leaves. Such a process can be constructed as follows: first, remove four edges so that the result is a $1 \times (n-1)$ grid graph. Now delete a top edge and the corresponding bottom edge in such a way as to break the graph into two components that are as nearly equal in size as possible; repeat this process till subgraphs are reached that consist of two nodes linked by a vertical line, then delete the vertical line. A branch in the resulting tree has length at most $2 \log n + c$, for some constant c, so that the tree has $O(n^2)$ leaves. □

Corollary 6.1 *The system G_{Tree} cannot p-simulate $Res(\equiv)_{Tree}$.*

In contrast to the foregoing results, if derivations are presented in linear form, then resolution and cut-free Gentzen systems are equally powerful systems (up to a polynomial) when pure biconditional tautologies are considered.

Theorem 6.4 *Each of the following systems can p-simulate any of the others: G, $G +$ analytic Cut, $Res(\equiv)$.*

Proof: It is trivially true that $G +$ *analytic Cut* can p-simulate G. In addition, the simulation of G_{Tree} by $Res(\equiv)_{Tree}$ in the previous section can be extended readily to a simulation of $G +$ *analytic Cut* by $Res(\equiv)$.

The simulation of G + *analytic Cut* by $Res(\equiv)$ can be reversed as follows. Given a refutation of $Def(A) \cup \{\overline{l_A}\}$ in $Res(\equiv)$, we can convert it into a shorter derivation of l_A in $Res(\equiv)$ by omitting any resolution involving the literal $\overline{l_A}$. We can then simulate this derivation in G + *analytic Cut* by using the reverse of the translation employed earlier; the analytic cut rule can be employed to simulate resolution inferences.

It remains only to show that G can p-simulate G + *analytic Cut* . We shall show how to replace an inference by analytic cut by a sequence of inferences using the inference rules of G so that the number of inferences of G used is a linear function of the length of the conclusion of the derivation. Thus let D be a derivation of a sequent $\Gamma \vdash \Delta$ in G + *analytic Cut* . Let

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma \vdash \Delta, A}{\Gamma \vdash \Delta}\ (Cut)$$

be an inference by the analytic Cut rule in D. The formula A is a subformula of a formula B occurring as an antecedent or consequent formula in the conclusion of the derivation. Thus there is a sequence of formulas $A = B_0 \cdots B_k = B$ so that B_i is an immediate subformula of B_{i+1}. By using the rules of G, we can derive $\Gamma, B_i \vdash \Delta$ and $\Gamma \vdash B_i, \Delta$ for any i. Thus, let us suppose that B_{i+1} has the form $(B_i \equiv C)$, and that we have already derived the sequents $\Gamma, B_i \vdash \Delta$ and $\Gamma \vdash B_i, \Delta$. By the weakening rule, we can derive $\Gamma, B_i, C \vdash \Delta$ and $\Gamma \vdash B_i, C, \Delta$ and so $\Gamma, B_{i+1} \vdash \Delta$ by $(\equiv \vdash)$. Symmetrically, we can derive $\Gamma \vdash B_{i+1}, \Delta$. This involves 6 extra steps in the proof, so it is possible to derive $\Gamma, B \vdash \Delta$ and $\Gamma \vdash B, \Delta$ in $6k$ extra steps. By repeating this manoeuvre, for any sequent $\Theta \vdash \Xi$ in the derivation D, we can derive a corresponding sequent in G of the form $\Theta, \Gamma' \vdash \Xi, \Delta'$, where Γ' and Δ' are subsets of Γ and Δ. The derivation of $\Gamma \vdash \Delta$ in G that results has complexity $O(k.m)$, where k is the complexity of D, and m is the number of symbols in $\Gamma \vdash \Delta$. $\square$

This somewhat unexpected simulation result depends on the special features of the inference rules for $\equiv$ in G. It extends easily to include negation, but does not appear to extend to conjunction and disjunction. Whether the simulation result holds when the cut-free Gentzen system includes these connectives is open.

We now sketch a result mentioned earlier, that the addition of the Thinning rule results in an exponential shortening of derivations in some cases.

Theorem 6.5 *A derivation of U_n in the system G' must contain at least $n.2^n$ distinct sequents.*

Proof: The proof of this result is essentially the same as the proof of Theorem 6.2. As in the earlier proof, we can trace an O-assignment back up a derivation of U_n to an axiom. Axioms corresponding to distinct O-assignments must be distinct, by the argument given earlier to show that the branches of the tree must be distinct. $\square$

Techniques similar to those used in the lower bound for resolution can be used to proved exponential lower bounds for cut-free Gentzen systems. The following result is proved in Urquhart [60].

Theorem 6.6 *There is a sequence F_n of biconditional tautologies, where each formula has length $O(n^2)$, but the shortest proof of F_n in G contains at least $2^{n/16}$ distinct sequents.*

This result can be improved to a lower bound exponential in the size of a family of biconditional tautologies based on expander graphs by adapting the proof of Theorem 5.3.

7 Frege systems

Proof systems p-equivalent to axiomatic systems with schematic axioms and rules form a natural and important class. This family of systems are called "Frege systems" in the literature of proof complexity. Strictly speaking, this is a misnomer, since Frege's original system of propositional logic [28] included a tacitly applied rule of substitution; according to Church [13, p. 158], Von Neumann [62] was the first to use axiom schemes to avoid the use of a substitution rule. However, the term "Frege system" seems well entrenched in the complexity literature, so it is employed here.

We assume in this and the following sections a language for propositional logic based on a functionally complete set of connectives, for example the language based on binary disjunction $\vee$ and negation $\sim$. We shall include in addition the propositional constants 0 and 1 standing for "false" and "true" respectively. As we shall see below, the exact choice of language is in many cases not crucial. If A is a formula and $p_1, \ldots, p_m$ a sequence of variables then we write $A[B_1/p_1, \ldots, B_m/p_m]$ for the formula resulting from A by substituting $B_1, \ldots, B_m$ for $p_1, \ldots, p_m$.

A *Frege rule* is defined to be a sequence of formulas written in the form $A_1, \ldots, A_k \vdash A_0$. In the case that the sequence $A_1, \ldots, A_k$ is empty, the rule is referred to as an *axiom scheme*. The rule is *sound* if $A_1, \ldots, A_k \models A_0$, that is, if every truth-value assignment satisfying $A_1, \ldots, A_k$ also satisfies A_0. If $A_1, \ldots, A_k \vdash A_0$ is a Frege rule, then C_0 is *inferred from* $C_1, \ldots, C_k$ by this rule if there is a sequence of formulas $B_1, \ldots, B_m$ and variables $p_1, \ldots, p_m$ so that for all i, $0 \leq i \leq k$, $C_i = A_i[B_1/p_1, \ldots, B_m/p_m]$.

If $\mathcal{F}$ is a set of Frege rules and A a formula, then a *proof of A in $\mathcal{F}$ from* $A_1, \ldots, A_m$ is a finite sequence of formulas such that every formula in the sequence is one of $A_1, \ldots, A_m$ or inferred from earlier formulas in the sequence by a rule in $\mathcal{F}$, and the last formula is A. The formulas in the sequence are the *lines* in the proof.

If $\mathcal{F}$ is a set of Frege rules, then it is *implicationally complete* if whenever $A_1, \ldots, A_m \models A_0$ then there is a proof of A_0 in $\mathcal{F}$ from $A_1, \ldots, A_m$. A *Frege* *system* is defined to be a finite set of sound Frege rules that is implicationally complete.

Example 7.1 *Shoenfield's system [53, p. 21], in which the primitive connectives are* $\vee$ *and* $\sim$*:*

Excluded middle: $\vdash {\sim} p \vee p$;

Expansion rule: $p \vdash q \vee p$;

Contraction rule: $p \vee p \vdash p$;

Associative rule: $p \vee (q \vee r) \vdash (p \vee q) \vee r$;

Cut rule: $p \vee q, {\sim} p \vee r \vdash q \vee r$.

We define the *size* of a Frege proof as the number of occurrences of symbols in it. Another measure is that of the number of lines in a proof; we shall refer to this measure as the *length* of a proof. The length and size measures of a proof may not be polynomially related, since it is possible for a Frege proof to contain lines that are exponentially large, as a function of the proof's length. The *complexity* of a Frege rule is the number of distinct formulas occurring in the rule; for example, the Cut rule in Shoenfield's system has complexity 7.

Many types of systems familiar in the logical literature are p-equivalent to Frege systems. Among these are systems obtained by adding the cut rule to cut-free Gentzen systems, and systems of natural deduction containing the deduction theorem as a primitive rule. The statement of this equivalence forms one of the main results in Reckhow's thesis [50].

Theorem 7.1 *Any two systems from the following classes are p-equivalent: Frege systems, natural deduction systems, Gentzen systems with cut.*

The proof of this theorem is straightforward when the two systems are based on the same connectives, or when there is a direct translation possible (for example, such a translation is possible between the connective sets $\{\rightarrow, \sim\}$ and $\{\vee, \sim\}$). However, a direct translation is not possible, for example, in the case of the connective sets $\{\rightarrow, \sim, \equiv\}$ and $\{\wedge, \sim\}$. In this case, it is necessary to employ a technique of indirect translation based on the well known construction of Spira [57], [64, pp. 218-221], [27, pp. 68-73]. The reader is referred to Reckhow [50] or the book by Krajíček [40] for details of the proof.

For general Frege systems only very weak lower bounds on the size of proofs are known. This failure in proving lower bounds mirrors the corresponding lack of success in proving strong lower bounds on the size of formulas or circuits computing explicitly defined Boolean functions (the books by Wegener and Dunne [27, 64] provide good surveys of work in this area). Strong lower bounds have been proved only in the case where significant restrictions are placed on the structure of the proofs considered. These restrictions are a counterpart in proof theory of restricted classes of circuits for which strong lower bounds are known.

We conclude this section with a sketch of the lower bounds just mentioned. We give a full outline of the proof, but omit the rather intricate details of the central combinatorial lemmas on which the proof rests. For the purpose of these lower bounds, we employ the language based on disjunction and negation, together with the constants 0 and 1; a conjunction $A \wedge B$ is treated as an abbreviation for the formula $\sim(\sim A \vee \sim B)$.

A formula can be represented by its formation tree in which the leaves are labeled with propositional variables or constants, and an interior node is labeled with $\vee$ if it is the parent of two nodes, and with $\sim$ if it is the parent of only one. A branch in the tree representing a formula, when traversed from the root to the leaf at the end of the branch is labeled with a block of operators of one kind (say $\sim$), followed by a block of the other kind (say $\vee$), ..., ending with a variable or constant. The *logical depth* of a branch is defined to be the number of blocks of operators labeling the branch. The *depth* of a formula is the maximum logical depth of the branches in its formation tree.

Example 7.2 *The formula* $(\sim p \vee \sim\sim 1) \vee \sim(\sim q \vee r)$ *has depth 4.*

The depth of a proof in a Frege system is the maximum depth of a line in the proof. The lower bound sketched below is for proofs of bounded depth, in which all formulas have depth bounded by a fixed constant.

The lower bound is based on the propositional pigeon-hole principle, mentioned above as the basis for Haken's exponential lower bound [35] for resolution. Let D, R be finite non-empty sets where $D \cap R = \emptyset$, and let $S = D \cup R$. A *matching* between D and R is a set of mutually disjoint unordered pairs $\{i, j\}$, where $i \in D$, $j \in R$ (that is to say, a matching in the complete bipartite graph $D \times R$). A matching *covers* a vertex i if $\{i, j\}$ belongs to the matching for some vertex j; a matching covers a set X if it covers all the vertices in X. If π is a matching then we denote by $V(\pi)$ the set of vertices covered by π. A matching between D and R is *perfect* if it covers all of the vertices in $D \cup R$.

The pigeon-hole principle states that if $|D| = n + 1$, $|R| = n$ then there is no perfect matching between D and R. To formalize this as a tautology in propositional logic we introduce propositional variablesP_{ij} for $i \in D$, $j \in R$. The language built from these variables and the constants 0,1 using the connectives $\vee, \sim$ we shall refer to as $L(D, R)$; we also refer to the language as L_n in contexts where D, R are understood as the basic sets. The tautology $PHP(D, R)$ is the disjunction

$$\bigvee_{\substack{i \neq j \in D \\ k \in R}} (P_{ik} \wedge P_{jk}) \vee \bigvee_{\substack{i \neq j \in R \\ k \in D}} (P_{ki} \wedge P_{kj}) \vee \bigvee_{i \in D} \bigwedge_{k \in R} \sim P_{ik} \vee \bigvee_{k \in R} \bigwedge_{i \in D} \sim P_{ik}.$$

We shall also refer to this as PHP_n when the underlying sets are understood. The negation of PHP_n is equivalent to the conjunction of a set of clauses; Haken

[35] showed that this set of clauses requires resolution refutations of size c^n, for $c > 1$.

The most important step so far in our understanding of the complexity of propositional proofs was taken by Ajtai in a remarkable paper [1] in which he proved the following result.

Theorem 7.2 *For a given Frege system $\mathcal{F}$, natural numbers c, d, and sufficiently large n, any depth c proof of PHP_n in $\mathcal{F}$ must have size greater than n^d.*

Ajtai's proof is a highly ingenious blend of non-standard models for number theory and combinatorics. Subsequent work by a number of authors simplified Ajtai's proof, first eliminating the use of non-standard models [9], second improving the lower bound from super-polynomial to exponential [39, 7, 47, 38]. Krajíček [39] proved the first truly exponential lower bounds for bounded depth proofs, using modified versions of the pigeon-hole formulas for each depth d. In the same paper, he also showed that depth d Frege systems cannot p-simulate depth $d+1$ Frege systems. Shortly afterwards, Pitassi, Beame and Impagliazzo [7, 47] and (independently) Krajíček, Pudlák and Woods [7, 38] improved Ajtai's lower bound for the pigeon-hole principle from super-polynomial to exponential; their proof is sketched here.

Let D, R be fixed, where $|D| = n+1$, $|R| = n$. The set of matchings between D and R we shall denote by M_n. A matching π determines a *restriction* ρ_π of the variables of L_n by the following definition. For a variable P_{ij}, if i or j is covered by π then $\rho_\pi(P_{ij}) = 1$ if $\{i,j\} \in \pi$, $\rho_\pi(P_{ij}) = 0$ if $\{i,j\} \notin \pi$; otherwise $\rho_\pi(P_{ij})$ is undefined. Since a matching uniquely determines and is determined by the corresponding restriction, we shall identify a matching with the restriction it determines, and refer to it according to context as a matching or a restriction. If ρ_1 and ρ_2 are two matchings in M_n, and $\rho_1 \cup \rho_2$ is also a matching, then we say that they are *compatible*. If ρ_1 and ρ_2 are compatible matchings, then their union will be written as $\rho_1\rho_2$. If ρ is a matching, then $D\restriction\rho = D \setminus V(\rho)$, $R\restriction\rho = R \setminus V(\rho)$ and $S\restriction\rho = S \setminus V(\rho)$. If M is a set of matchings, and ρ a matching, then $M\restriction\rho$ is defined to be $\{\rho' \setminus \rho : \rho' \in M,\ \rho' \text{ compatible with } \rho\}$.

If A is a formula of L_n, and $\rho \in M_n$, then we denote by $A\restriction\rho$ the formula resulting from A by substituting for the variables in A the constants representing their value under ρ. That is to say, if P_{ij} is set to 1 or 0 by ρ, then we substitute 1 or 0 for P_{ij}, otherwise the variable is unchanged. If Γ is a set of formulas and $\rho \in M_n$ then $\Gamma\restriction\rho$ is $\{A\restriction\rho : A \in \Gamma\}$. The formula $A\restriction\rho$ can be simplified by eliminating the constants by the rules $\neg 0 \equiv 1$, $\neg 1 \equiv 0$, $(0 \vee A) \equiv A$, $(A \vee 0) \equiv A$, $(1 \vee A) \equiv 1$, $(A \vee 1) \equiv 1$. If a formula A can be simplified to a formula B using these rules, then we write $A \equiv B$.

The language L_n contains only binary disjunction. However, in the proofs that follow it is convenient to introduce an auxiliary language that uses unbounded conjunctions and disjunctions. We shall distinguish the order of the terms in such conjunctions and disjunctions.

Let A be an unbounded conjunction each of whose conjuncts is a variable of L_n or a constant. We shall say that A is a *matching term* if the set of pairs $\{i,j\}$ for P_{ij} a variable in A forms a matching. The *size* of a matching term is the cardinality $|\pi|$ of the matching π corresponding to it; the set of vertices $V(A)$ associated with a matching term A is the set of vertices mentioned in the variables in A, that is, the set $V(\pi)$. If π is a matching, then we shall write $\wedge\pi$ for the matching term that describes it, the conjunction containing the set of variables P_{ij} for $\{i,j\} \in \pi$ as conjuncts.

An unbounded disjunction of matching terms we shall call a *matching disjunction*; it is a *matching disjunction over S* if all the vertices mentioned in it are in S. If all of the matching terms in a matching disjunction have size bounded by r, then it is an *r-disjunction*.

Let A be a disjunction in the language L_n, and A_i, $i \in I$, those subformulas of A that are not disjunctions, but every subformula of A properly containing them is a disjunction. Then the *merged form of A* is the unbounded disjunction $\bigvee_{i \in I} A_i$.

The proof of the lower bound is significantly complicated in this case by the fact that we are dealing with a system in which all the steps are tautologies. In contrast, the lower bound for resolution (for example) exploits the fact that a refutation can be considered as making progress towards a contradiction. It is plain that to have similar notion of "progress" in this case, we have to employ a non-standard "truth" definition. The solution to this problem is to assign each step in a derivation its own space of assignments with respect to which it is a "tautology"; if we choose the spaces in the right way, the rules of inference are sound with respect to these "local tautologies."

The spaces of local assignments are provided by matching trees, that is, decision trees in which the branches represent matchings. We assume that the space of matchings is the set M_n of matchings between D and R, where $|D| = n+1$, $|R| = n$, $S = D \cup R$.

Definition 7.1 *A* matching tree *over S is a tree T satisfying the conditions:*

1. *The nodes of T other than the leaves are labeled with vertices in S;*
2. *If a node in T is labeled with a vertex $i \in S$, then the edges leading out of the node are labeled with distinct pairs of the form $\{i,j\}$ where $j \in R$ if $i \in D$, $j \in D$ if $i \in R$;*
3. *No node or edge label is repeated on a branch of T;*
4. *If p is a node of T then the edge labels on the path from the root of T to p determine a matching $\pi(p)$ between D and R.*

We shall use the notation $Br(T)$ for the set of matchings determined by the branches of T, that is, $\{\pi(l) : l \text{ a leaf in } T\}$. If M is a set of matchings, then T is said to be *complete for M* if for any node p in T labeled with a vertex

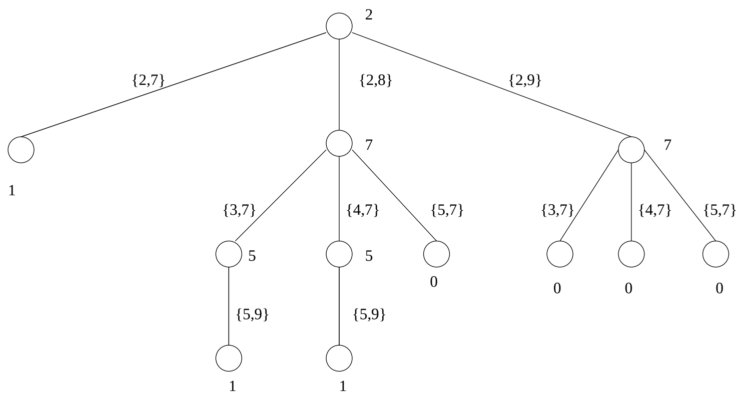

Figure 4: A matching decision tree

$i \in S$, the set of matchings $\{\pi(q) : q \text{ a child of } p\}$ consists of all matchings in M of the form $\pi(p) \cup \{\{i, j\}\}$. If the space of matchings is M_n, we shall use the abbreviation "complete" instead of "complete for M_n."

Lemma 7.1 *Let T be a complete matching tree over the space $S = D \cup R$, $|D| = n + 1$, $|R| = n$, and ρ a matching in M_n such that $|\rho| + Depth(T) \leq n$. Then there is a $\pi \in Br(T)$ such that $\pi \cup \rho \in M_n$.*

Proof. We show that by successively choosing nodes in T starting at the root we can find a branch in T so that the required π labels the chosen path. Let us suppose that the nodes have been chosen as far as a node p that is not a leaf. By assumption, $\rho \cup \pi(p) \in M_n$; since $|\rho| + Depth(T) \leq n$, $|\rho \cup \pi(p)| < n$. Let i be the vertex in S labeling node p; there exists at least one matching extending $\rho \cup \pi(p)$ that covers i. Since T is complete, at least one edge below p is labeled with a pair that extends $\rho \cup \pi(p)$ to a matching in M_n. Then we can extend the path by choosing the node at the end of this edge. $\square$

If the leaves of a matching tree T are each labeled with 0 or 1, then it is a *matching decision tree.* We define for $i = 0, 1$,

$$Br_i(T) = \{\pi(l) : l \text{ is a leaf of } T \text{ labeled } i\}.$$

If T is a matching decision tree, then T^c is the matching decision tree that results by changing the leaf labels of T from 0 to 1 and 1 to 0, while $Disj(T)$ is the unbounded disjunction $\bigvee\{\wedge\pi : \pi \in Br_1(T)\}$. Figure 4 shows a matching decision tree where $D = \{1, 2, 3, 4, 5\}$ and $R = \{6, 7, 8, 9\}$.

Lemma 7.2 *If T is a matching decision tree, and ρ extends a matching $\pi(l) \in Br(T)$, then $Disj(T)\lceil \rho \equiv 0$ or 1 according to whether l is labeled 0 or 1.*

Proof. If l is labeled 1, then since ρ extends $\pi(l)$, the term $\wedge\pi(l)$ is set to 1 by ρ, so that $Disj(T)\restriction\rho \equiv 1$. If l is labeled 0, then we need to establish that for any leaf l' labeled 1, $\wedge\pi(l')\restriction\rho \equiv 0$. Let p be the node at which the branches ending in l and l' diverge. If i is the vertex in S labeling p, then $\pi(l)$ and $\pi(l')$ must disagree on the vertex matched with i. Thus $\wedge\pi(l')\restriction\rho \equiv 0$, showing that $Disj(T)\restriction\rho \equiv 0$. □

If F is a matching disjunction, and T a matching decision tree, then we say that T *represents* F if for every $\pi(l) \in Br(T)$, $F\restriction\pi(l) \equiv 1$ if l is labeled 1, and $F\restriction\pi(l) \equiv 0$ if l is labeled 0.

We now introduce the basic concept of a k-evaluation; a k-evaluation can be considered as a kind of non-standard truth-definition for a set of formulas. The notion of k-evaluation is due to Krajíček, Pudlák and Woods [38]. The definition of k-evaluation used here differs from that of [38]; in that paper a more general definition is used in which formulas are assigned sets of restrictions rather than complete decision trees.

Definition 7.2 *Let Γ be a set of formulas of L_n, closed under subformulas, where $S = D \cup R$, $|D| = n+1$, $|R| = n$. Let $k > 0$. A k-evaluation T is an assignment of complete matching decision trees $T(A)$ to formulas $A \in L_n$ so that:*

1. *$T(A)$ has depth $\leq k$;*
2. *$T(1)$ is the tree with a single node labeled 1, and $T(0)$ is a tree of depth ≤ 1 with all its leaves labeled 0;*
3. *$T(P_{ij})$ is the full matching tree for $\{i,j\}$ over S, with a leaf l labeled 1 if $\pi(l)$ contains $\{i,j\}$, otherwise 0;*
4. *$T(\neg A) = T(A)^c$;*
5. *If A is a disjunction, and $\bigvee_{i\in I} A_i$ is the merged form of A then $T(A)$ represents $\bigvee_{i\in I} Disj(T(A_i))$.*

If T is a k-evaluation for a set of formulas Γ, then the set of matchings $Br(T(A))$ can be considered as a space of truth-value assignments for A; thus if $T(A)$ has all its leaves labeled 1, we can think of A as a kind of "tautology" relative to this space. However, in contrast to the classical notion of tautology, this notion is not preserved under classically sound inferences (this fact is the key to the lower bound argument).

Example 7.3 *Let $D = \{1,2,3\}$ and $R = \{4,5\}$, and let $\Gamma = \{P_{14} \vee P_{15}, \neg P_{15} \vee \neg P_{25}, P_{14} \vee \neg P_{25}\}$. Then there is a 2-evaluation for Γ so that the first two formulas in Γ have 1 on all their leaves, but the third formula does not, although it is a logical consequence of the first two.*

The following lemma shows that examples like this do not exist if the depth of a k-evaluation is small enough relative to the size of the inference rules of the proof system.

Lemma 7.3 *Let $\mathcal{F}$ be a Frege system in which the complexity of the rules is bounded by f, and $\mathcal{P}$ a proof in $\mathcal{F}$ in the language $L(D,R)$, where $S = D \cup R$, $|R| = n$. If T is a k-evaluation for all the formulas in $\mathcal{P}$ and $k \leq n/f$, then for any line A in $\mathcal{P}$,*

$$\forall \pi (\pi \in Br(T(A)) \Rightarrow Disj(T(A)) \restriction \pi \equiv 1),$$

that is, $T(A)$ has all of its leaves labeled 1.

Proof. The lemma is proved by induction on the number of lines in the proof $\mathcal{P}$. Let

$$\frac{A_1(B_1/p_1,\ldots,B_m/p_m),\ldots,A_k(B_1/p_1,\ldots,B_m/p_m))}{A_0(B_1/p_1,\ldots,B_m/p_m)}$$

be an instance of a rule of $\mathcal{F}$, and assume that the lemma holds for all of the premisses of the inference. Let Γ be the set of formulas $A(B_1/p_1,\ldots,B_m/p_m)$, where $A(p_1,\ldots,p_m)$ is a subformula of some A_i. By assumption, $|\Gamma| \leq f$; let $M = \{\pi_1 \cup \ldots \cup \pi_j \in M_n : \pi_i \in Br(T(C_i))\}$, where $\Gamma = \{C_1,\ldots,C_j\}$. By Lemma 7.1, if $\pi_i \in Br(T(C_i))$, then there is a $\pi \in M_n$ so that $\pi_i \subseteq \pi$. Let us abbreviate $Disj(T(A))$ as $D(A)$. Then for $\pi \in M$ and $A, B \in \Gamma$,

1. $D(A)\restriction \pi \equiv 0$ or $D(A)\restriction \pi \equiv 1$;
2. $D(0)\restriction \pi \equiv 0$ and $D(1)\restriction \pi \equiv 1$;
3. If $\neg A \in \Gamma$ then $D(\neg A)\restriction \pi \equiv 1 \Leftrightarrow D(A)\restriction \pi \equiv 0$;
4. If $(A \vee B) \in \Gamma$ then $D(A \vee B)\restriction \pi \equiv 1 \Leftrightarrow D(A)\restriction \pi \equiv 1$ or $D(B)\restriction \pi \equiv 1$.

These equivalences follow from the definition of a k-evaluation and from Lemmas 7.1 and 7.2.

For any $\pi \in M$, define an assignment V_π of truth-values to the formulas in Γ by setting $V_\pi(C_i) = 1$ if $D(C_i)\restriction \pi \equiv 1$, $V_\pi(C_i) \equiv 0$ if $D(C_i)\restriction \pi \equiv 0$. The list of equivalences above shows that V_π respects the rules of classical logic. By Lemma 7.2, the premisses of the inference are all assigned the value 1 by V_π; since the rule of inference is sound, the conclusion of the inference is also assigned 1 by V_π. Now let $\sigma \in Br(T(A_0(B_1/p_1,\ldots,B_m/p_m)))$. There is a $\pi \in M$ extending σ, so $V_\pi(A_0(B_1/p_1,\ldots,B_m/p_m)) = 1$, equivalently, $D(A_0(B_1/p_1,\ldots,B_m/p_m))\restriction \sigma \equiv 1$, concluding the proof of the lemma. $\square$

The next lemma shows that, relative to a k-evaluation, $k < n-1$, the pigeon-hole tautology PHP_n is a "contradiction."

Lemma 7.4 *Let $D \cup R = S$, $|D| = n+1$, $|R| = n$, $PHP_n = PHP(D, R)$. If T is a k-evaluation for a set of formulas containing PHP_n, $k < n-1$, then all the leaves of $T(PHP_n)$ are labeled 0.*

Proof. Left as an exercise for the reader; the proof uses Lemma 7.2. □

We now state without proof the central lemma showing that if a set of bounded depth formulas of L_n is subjected to a random restriction then, provided the set is not too large, the set of restricted formulas has associated decision trees of small depth. From this result the lower bound on the size of propositional proofs follows by earlier lemmas.

Lemma 7.5 *Let d be an integer, $0 < \epsilon < 1/5$, $0 < \delta < \epsilon^d$ and Γ a set of formulas of L_n of depth $\leq d$, closed under subformulas. If $|\Gamma| < 2^{n^\delta}$, $q = \lceil n^{\epsilon^d} \rceil$ and n is sufficiently large, then there exists $\rho \in M_n^q$ so that there is a $2n^\delta$-evaluation of $\Gamma \lceil \rho$.*

This lemma is proved by induction on the depth d. The induction step is handled by a "switching lemma" that says (roughly speaking) that if a matching disjunction is simplified by a random restriction, then with high probability the resulting simplified disjunction can be represented by a small depth decision tree. The name "switching lemma" derives from the corresponding combinatorial lemmas in circuit theory [29, 34], showing that with high probability, the application of a random restriction makes it possible to switch efficiently between conjunctive and disjunctive normal form ("efficiently" in the sense that a large blow-up in formula size does not occur). These lemmas allow the proof of strong lower bounds on the size of bounded depth circuits computing functions such as parity. The reader is referred to papers of Razborov [48] and Beame [5] for elegant proofs of various switching lemmas.

Theorem 7.3 *Let $\mathcal{F}$ be a Frege system and $d > 4$. Then for sufficiently large n every depth d proof in $\mathcal{F}$ of PHP_n must have size at least 2^{n^δ}, for $\delta < (1/5)^d$.*

Proof. Let the rules of $\mathcal{F}$ have complexity bounded by f, $0 < \delta < (1/5)^d$, and let $A_1, \ldots, A_t$ be a proof in $\mathcal{F}$ of depth d and size $\leq 2^{n^\delta}$.

Choose ϵ so that $\epsilon < 1/5$, $\delta < \epsilon^d$. By Lemma 7.5, there exists $\rho \in M_n^q$, $q = \lceil n^{\epsilon^d} \rceil$, and a $2n^\delta$-evaluation T of $\Gamma \lceil \rho$, where Γ is the set of subformulas in the proof $A_1, \ldots, A_t$. Then $A_1 \lceil \rho, \ldots, A_t \lceil \rho$ is a proof in $\mathcal{F}$ in the language $L(D \lceil \rho, R \lceil \rho)$.

Since $\delta < \epsilon^d$ and n is sufficiently large, $2n^\delta \leq n^{\epsilon^d}/f$, so by Lemma 7.3, for every step A_k in the proof, $T(A_k \lceil \rho)$ has all its leaves labeled 1. On the other hand, $PHP_n \lceil \rho \equiv PHP(D \lceil \rho, R \lceil \rho)$, so by Lemma 7.4, if PHP_n were the last line A_t of the proof, all the leaves of $T(PHP_n \lceil \rho)$ would be labeled 0. It follows that $A_1, \ldots A_t$ cannot be a proof of PHP_n. Hence, any proof in $\mathcal{F}$ of PHP_n must have size at least 2^{n^δ}. □

In a subsequent paper [2], Ajtai extended his lower bound to a system obtained from a bounded depth Frege system by adding certain axiom schemes. The pigeonhole principle PHP_n states that there is no perfect matching in the bipartite graph $D \times R$, where $|D| = n + 1$, and $|R| = n$. Let PAR_n be the tautology defined in a similar way stating that there is no perfect matching in the complete graph K_{2n+1}. Ajtai proves that even when we add to a Frege system all of the pigeonhole formulas PHP_n as new axiom schemes, the tautologies PAR_n still require bounded depth proofs that grow faster than any polynomial in n, when the proofs are restricted to a fixed depth (the pigeonhole formulas can be derived very easily by proofs of bounded depth when the formulas PAR_n are taken as axiom schemes). Beame and Pitassi [8] recently extended Ajtai's result by showing an exponential lower bound on the size of bounded depth Frege proofs of PAR_n in Frege systems with the added pigeonhole schemes.

Ajtai provided a further extension of these results in recent work [3, 4] on the independence of modular counting principles. The modulo q counting principle states that no finite set whose cardinality is not divisible by q can be partitioned into q-element classes. For a fixed cardinal number N, this principle can be stated as a propositional tautology $Count_q^N$; in this notation, the principle PAR_n can be expressed as $Count_2^{2n+1}$. Ajtai proved that whenever p, q are distinct primes, the propositional formulas $Count_q^{qn+1}$ do not have polynomial size, bounded depth Frege proofs from instances of $Count_p^m$, where $m \not\equiv 0 \pmod{p}$. Beame, Impagliazzo, Krajíček, Pitassi and Pudlák [6] extended this result to composite p and q.

The preceding results are significant not just from the point of view of propositional complexity theory, but also as providing independence results in systems of bounded arithmetic. The system $I\Delta_0$ of first order bounded arithmetic has been extensively studied; in it, the induction scheme is restricted to formulas containing only bounded quantifiers. Let $I\Delta_0(f)$ be the system obtained from $I\Delta_0$ by adding a new function symbol f that is allowed to appear in the induction scheme. Let $PHP(f)$ be the formula in the expanded language expressing the fact that f is not a bijection between $\{0, \ldots, n\}$ and $\{1, \ldots, n\}$ for any n. Then Ajtai's lower bound for bounded depth proofs shows that $PHP(f)$ is unprovable in $I\Delta_0(f)$; similar independence results can be proved for the modular counting principles. This follows from the fact that for statements S of an appropriate syntactic form, there is a corresponding sequence of tautologies expressing restricted versions of S so that if S is provable in bounded arithmetic, the sequence of tautologies has polynomial-size proofs in a bounded-depth Frege system. The reader is referred to a paper by Buss [11] and the book by Krajíček [40] for details of this connection.

8 The extension and substitution rules

A natural way to extend a Frege system is to allow the possibility of abbreviating formulas by definitions. This idea was first proposed by Tseitin [59] in the context of resolution, but is perhaps more natural in the context of axiomatic systems.

Let $\mathcal{F}$ be a Frege system; for convenience, let us assume that the language of $\mathcal{F}$ contains a symbol $\equiv$ for the biconditional. If $\Gamma \cup \{A\}$ is a set of formulas of $\mathcal{F}$, then a sequence of formulas ending in A is a *proof of A from Γ in $\mathcal{F}$ with extension* if each formula in the sequence either belongs to Γ, or is inferred from earlier formulas in the sequence by one of the rules of $\mathcal{F}$ or has the form $P \equiv A$, where P is a variable not in appearing in $\Gamma \cup \{A\}$, nor in any earlier formula in the sequence. In the case of a step of the last type, the variable is said to be introduced by the *extension rule*. We shall refer to the system with the addition of the extension rule as an *extended Frege system*. The extension rule appears to be very powerful; since abbreviations can be iterated, very long formulas can be abbreviated to short ones by using the extension rule.

The substitution rule is another natural rule that appears in the earliest systems for propositional logic, such as those of Frege [28] and Whitehead and Russell [65]. (Since the substitution rule is unsound, in proofs from assumptions, substitution must be disallowed in variables appearing in assumptions.) It is not hard to prove that a Frege system with the addition of the substitution rule can p-simulate the same system with the extension rule added. Surprisingly, the converse simulation also holds, a result first proved by Dowd [26].

Theorem 8.1 *Any two systems from the following classes are p-equivalent: extended resolution, extended Frege systems, Frege systems with substitution.*

Proof. See Krajíček and Pudlák [41]. □

Extended Frege systems are significant in themselves as a natural class of proof systems, but also because of a connection with another form of bounded arithmetic. This was the first connection to be observed between propositional logic and bounded arithmetic; it appeared in a fundamental paper of Cook [18]. The system PV is a free variable system of arithmetic that bears the same relation to the polynomial-time computable functions as Skolem's recursive arithmetic bears to the primitive recursive functions. Whereas Skolem's system has a function symbol for each primitive recursive function, PV has one for each polynomial-time computable function; for details, see [21].

Cook introduced PV as a way of formalizing the intuitive notion of 'feasibly constructive proof': feasibly constructive proof is to polynomial-time algorithm as constructive proof is to algorithm. The next theorem (from Cook [18]) emphasizes the power of extended Frege systems; it shows that if a combinatorial principle has a feasibly constructive proof, then the corresponding family of tautologies has polynomial-size proofs in an extended Frege system.

Theorem 8.2 *If $t = u$ is an equation of PV then there is a polynomially growing family of propositional formulas $|t = u|_n$ so that:*

1. *The formula $|t = u|_n$ is a tautology iff $t = u$ is true when restricted to numerals of length n or less;*

2. *If $\vdash_{PV}$ $t = u$ then there is a polynomial $p(n)$ so that $|t = u|_n$ has an extended Frege proof of length at most $p(n)$ for all n.*

Proof. A detailed proof is contained in Dowd's thesis [25]. □

The next theorem shows that an extended Frege system can p-simulate any proof system whose soundness is provable in PV. In Section 2, a proof system was defined as a polynomial-time computable function; thus any proof system is represented by a primitive function symbol of PV. In particular, let EF be the function symbol of PV representing a fixed extended Frege system. Let $TRUE(x, y)$ be the arithmetical function with range $\{0, 1\}$ such that $TRUE(m, n) = 1$ if and only if m is the encoding of a propositional formula A and n the encoding of an assignment under which A is true. If P is a function symbol of PV representing a proof system, then the soundness of P can be expressed in the form $\forall x \forall y[TRUE(P(x), y) = 1]$.

Theorem 8.3 *If $PV \vdash TRUE(P(x), y) = 1$, then there is a function symbol G of PV so that $PV \vdash EF(G(x)) = P(x)$.*

This is the main result of Cook [18]. A detailed proof is in Dowd [25]. Combined with Dowd's observation that the soundness of a Frege system with substitution is provable in an extended Frege system, it leads immediately to Theorem 8.1.

The foregoing results all emphasize the power of extended Frege systems, and tend to show that proving lower bounds for such systems is a formidable challenge. There are also some theoretical reasons to think that such results will be hard to obtain. Cook and Urquhart [21] show that in a precisely defined sense there can be no feasibly constructive proof of a super-polynomial lower bound for an extended Frege system. Buss [12] and Krajíček and Pudlák [42] have proved further results along the same lines.

We conclude this survey of proof systems by mentioning the quantified propositional calculus, the form of second order logic obtained by adding rules for propositional quantifiers to a Frege system [13, §28]. Such systems can simulate Frege systems with substitution, but otherwise little is known about their complexity. There are significant connections between complexity questions about such systems and problems in bounded arithmetic. The reader is referred to the work of Dowd [25] and Krajíček and Pudlák [43].

It is not known whether in the p-simulation ordering there is a greatest element, that is, whether or not there is a propositional proof system that p-simulates all propositional proof systems. Krajíček and Pudlák [41] discuss the relation of this question to other well known open problems in computational complexity theory.

9 Open Problems

The major questions in the area of the complexity of propositional proofs remain unsolved. Of these, perhaps the most important and challenging is that of proving super-polynomial lower bounds on the length of proofs in Frege and extended Frege systems. Even substantial improvements on the currently known weak lower bounds would be of considerable interest.

The best known lower bound on the size of Frege proofs is quadratic. It rests on the observation that in a Frege proof, each application of a schematic rule can involve only a finite number of "active" subformulas. Hence, in a Frege proof of a tautology that is not a substitution instance of a smaller tautology, all the subformulas must occur as active subformulas somewhere in the proof. (For the details of this result, see Krajíček [40, Ch. 4].)

Problem 9.1 *Prove a lower bound on the size of Frege proofs that is better than quadratic.*

The next problem is probably not too difficult, but might require a new idea.

Problem 9.2 *Can a cut-free Gentzen system based on the connectives* $\{\vee, \sim\}$ *p-simulate resolution as a system for refuting contradictory sets of clauses?*

A natural extension of the results on bounded depth Frege proofs would be to prove lower bounds for proofs of bounded depth where we allow not only unbounded disjunctions and conjunctions, but also unbounded connectives computing the parity function. That is to say, we alter the definition of depth above to allow unbounded logical gates $(x_1 \oplus \ldots \oplus x_n)$ computing addition modulo 2 to count as formulas of depth 1. Strong lower bounds have been proved by Razborov and Smolensky using the corresponding model of bounded depth circuits [49, 55].

Problem 9.3 *Can we prove superpolynomial lower bounds on the complexity of bounded depth Frege proofs, using the modified definition of depth?*

References

[1] Miklós Ajtai. The complexity of the pigeonhole principle. In *Proceedings of the 29th Annual IEEE Symposium on the Foundations of Computer Science*, pages 346–355, 1988.

[2] Miklós Ajtai. Parity and the pigeonhole principle. In Samuel R. Buss and Philip J. Scott, editors, *Feasible Mathematics*, pages 1–24. Birkhäuser, 1990.

[3] Miklós Ajtai. The independence of the modulo p counting principles. Preprint, 1993.

[4] Miklós Ajtai. Symmetric systems of linear equations modulo p. Preprint, 1993.

[5] Paul Beame. A switching lemma primer. Preprint, 1993.

[6] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower bounds on Hilbert's Nullstellensatz and propositional proofs. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 794–806. IEEE Computer Society Press, 1994.

[7] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, Pavel Pudlák, and Alan Woods. Exponential lower bounds for the pigeonhole principle. In *Proceedings of the 24th Annual ACM Symposium on the theory of computing*, pages 200–220, 1992.

[8] Paul Beame and Toniann Pitassi. An exponential separation between the matching principles and the pigeonhole principle. Forthcoming, *Annals of Pure and Applied Logic*, 1993.

[9] Stephen Bellantoni, Toniann Pitassi, and Alasdair Urquhart. Approximation and small-depth Frege proofs. *SIAM Journal of Computing*, 21:1161–1179, 1992.

[10] Samuel R. Buss. *Bounded Arithmetic*. Bibliopolis, Naples, 1986.

[11] Samuel R. Buss. Polynomial size proofs of the propositional pigeonhole principle. *Journal of Symbolic Logic*, 52:916–927, 1987.

[12] Samuel R. Buss. On model theory for intuitionistic bounded arithmetic with applications to independence results. In Samuel R. Buss and Philip J. Scott, editors, *Feasible Mathematics*, pages 27–47. Birkhäuser, 1990.

[13] Alonzo Church. *Introduction to Mathematical Logic*. Princeton U. P., 1956.

[14] Vašek Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25:285–287, 1979.

[15] Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the Association for Computing Machinery*, 35:759–768, 1988.

[16] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on the Theory of Computation*, pages 151–158, 1971.

[17] Stephen A. Cook. An exponential example for analytic tableaux. Manuscript, 1973.

[18] Stephen A. Cook. Feasibly constructive proofs and the propositional calculus. In *Proceedings of the Seventh Annual ACM Symposium on the Theory of Computation*, pages 83–97, 1975.

[19] Stephen A. Cook and Robert A. Reckhow. On the lengths of proofs in the propositional calculus. In *Proceedings of the Sixth Annual ACM Symposium on the Theory of Computing*, 1974. See also corrections for above in *SIGACT News*, Vol. 6 (1974), pp. 15-22.

[20] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44:36–50, 1979.

[21] Stephen A. Cook and Alasdair Urquhart. Functional interpretations of feasibly constructive arithmetic. *Annals of Pure and Applied Logic*, 63:103–200, 1993.

[22] Marcello D'Agostino. Are tableaux an improvement on truth-tables? *Journal of Logic, Language and Information*, 1:235–252, 1992.

[23] Martin Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the Association for Computing Machinery*, 5:394–397, 1962. Reprinted in [54], Vol. 1, pp. 267-270.

[24] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7:201–215, 1960. Reprinted in [54], Vol. 1, pp. 125-139.

[25] Martin Dowd. *Propositional representation of arithmetic proofs*. PhD thesis, University of Toronto, 1979. Department of Computer Science, Technical Report No. 132/79.

[26] Martin Dowd. Model-theoretic aspects of $\mathcal{P} \neq \mathcal{NP}$. Unpublished MS, 1985.

[27] Paul E. Dunne. *The Complexity of Boolean networks*. Academic Press, London and San Diego, 1988.

[28] Gottlob Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Nebert, Halle, 1879.

[29] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. In *Proceedings of the 22nd Annual IEEE Symposium on the Foundations of Computer science*, pages 260–270, 1981.

[30] Ofer Gabber and Zvi Galil. Explicit constructions of linear size superconcentrators. In *Proceedings 20th Annual Symposium on Foundations of Computer Science*, pages 364–370, New York, 1979. IEEE.

[31] Zvi Galil. On the complexity of regular resolution and the Davis-Putnam procedure. *Theoretical Computer Science*, 4:23–46, 1977.

[32] Michael R. Garey and David S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-completeness.* W.H. Freeman, 1979.

[33] Andreas Goerdt. Comparing the complexity of regular and unrestricted resolution. In *Proceedings of the 14th German Workshop on A.I.* Informatik Fachberichte 251, 1990.

[34] Johan T. Håstad. *Computational Limitations of Small-Depth Circuits.* MIT Press, 1987.

[35] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[36] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[37] R. Kowalski and P. J. Hayes. Semantic trees in automatic theorem-proving. In Meltzer and Michie, editors, *Machine Intelligence Vol. 4*, pages 87–101. Edinburgh U. Press, Edinburgh, 1969.

[38] Jan Krajíček, Pavel Pudlák, and Alan Woods. Exponential lower bound to the size of bounded depth Frege proofs of the pigeonhole principle. Submitted, 1991.

[39] Jan Krajíček. Lower bounds to the size of constant-depth propositional proofs. *Journal of Symbolic Logic*, 59:73–86, 1994.

[40] Jan Krajíček. *Bounded Arithmetic, Propositional Logic and Complexity Theory.* Cambridge University Press, 1996.

[41] Jan Krajíček and Pavel Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *Journal of Symbolic Logic*, 54:1063–1079, 1989.

[42] Jan Krajíček and Pavel Pudlák. Propositional provability in models of weak arithmetic. In E. Börger, H. Kleine-Bunning, and M.M. Richter, editors, *Computer Science Logic (Kaiserlautern, Oct. '89)*, pages 193–210. Springer-Verlag, 1990. LNCS 440.

[43] Jan Krajíček and Pavel Pudlák. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 36:29–46, 1990.

[44] G. A. Margulis. Explicit construction of concentrators. *Problems of Information Transmission*, 9:325–332, 1973.

[45] Neil V. Murray and Erik Rosenthal. On the computational intractability of analytic tableau methods. *Bulletin of the IGPL*, Volume 2, Number 2:205–228, September 1994.

[46] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[47] Toniann Pitassi, Paul Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3:97–140, 1993.

[48] Alexander A. Razborov. Bounded arithmetic and lower bounds in Boolean complexity. Manuscript 1993, submitted to the volume *The mathematics of Paul Erdös*.

[49] Alexander A. Razborov. Lower bounds on the size of bounded depth networks over a complete basis with logical addition. *Matemat. Zametki*, 41:598–607, 1987. English translation in: *Mathematical Notes*, Vol. 41 (1987), 333-338.

[50] Robert Reckhow. *On the lengths of proofs in the propositional calculus*. PhD thesis, University of Toronto, 1976.

[51] J. A. Robinson. The generalized resolution principle. In Dale and Michie, editors, *Machine Intelligence, Vol. 3*, pages 77–94. American Elsevier, New York, 1968. Reprinted in [54], Vol. 2, pp. 135-151.

[52] N.A. Shanin, G.V. Davydov, S. Y. Maslov, G.E. Mints, V.P. Orevkov, and A.O. Slisenko. *An algorithm for a machine search of a natural logical deduction in a propositional calculus*. Izdat. Nauka, Moscow, 1965. Reprinted in [54].

[53] Joseph Shoenfield. *Mathematical Logic*. Addison-Wesley, 1967.

[54] Jörg Siekmann and Graham Wrightson, editors. *Automation of Reasoning*. Springer-Verlag, New York, 1983.

[55] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on the Theory of Computing*, pages 77–82, 1987.

[56] Raymond M. Smullyan. *First-order Logic*. Springer-Verlag, New York, 1968. Reprinted by Dover, New York, 1995.

[57] P.M. Spira. On time-hardware complexity tradeoffs for Boolean functions. In *Proceedings of the fourth Hawaii International Symposium on System Sciences*, pages 525–527, 1971.

[58] Richard Statman. Bounds for proof-search and speed-up in the predicate calculus. *Annals of mathematical logic*, 15:225–287, 1978.

[59] G.S. Tseitin. On the complexity of derivation in propositional calculus. In A. O. Slisenko, editor, *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, pages 115–125. Consultants Bureau, New York, 1970. Reprinted in [54], Vol. 2, pp. 466-483.

[60] Alasdair Urquhart. The complexity of Gentzen systems for propositional logic. *Theoretical Computer Science*, 66:87–97, 1989.

[61] Alasdair Urquhart. The relative complexity of resolution and cut-free gentzen systems. *Annals of mathematics and artificial intelligence*, 6:157–168, 1992.

[62] John von Neumann. Zur Hilbertschen Beweistheorie. *Mathematische Zeitschrift*, 26:1–46, 1926.

[63] Hao Wang. Towards mechanical mathematics. *IBM Journal for Research and Development*, 4:2–22, 1960. Reprinted in [54], Vol. 1, pp. 244-264.

[64] Ingo Wegener. *The Complexity of Boolean Functions.* B.G. Teubner and John Wiley, 1987.

[65] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica, Vols. 1-3.* Cambridge University Press, 1910-1913. Second edition, 1925.

University of Toronto
Toronto, Ontario