On the Complexity of Resolution Procedures for Theorem Proving

by Zvi Galil

TR 74-223

December 1974

Department of Computer Science
Cornell University
Ithaca, N.Y. 14853

On the Complexity of Resolution

Procedures for Theorem Proving

Zvi Galil †

Department of Computer Science

Cornell University

Ithaca, N.Y.

Abstract

We study several procedures for theorem proving based on the resolution principle. We consider (1) Davis Putnam procedure; (2) regular resolution; (3) unrestricted resolution; (4) resolution with extension; and (5) several versions of bounded resolution. The powers of these procedures are compared. Exponential lower bounds are proved for the run-time of some of them.

† This research was partially supported by ONR Grant N00014-67-A-0077-002:

On the Complexity of Resolution Procedures for Theorem Proving

Zvi Galil

1. Introduction

One important problem in computer science is the "tautology problem": given a boolean formula in disjunctive normal form, how does one determine whether it is a tautology? (i.e., whether it is true for any truth assignment to its variables). The tautology problem is of central importance in computer science, since

1) [1] There is a polynomial time algorithm for the tautology problem iff the class of problems solved by polynomial time bounded Turing machines (P) is equal to the class of problems solved by nondeterministic polynomial time bounded Turing machines (NP); and

2) [3] There is a nondeterministic polynomial time bounded algorithm for tautology iff NP is closed under complementation.

Thus, the tautology problem is closely related to two of the most important open problems in the theory of complexity of computing. The dual problem of the tautology problem is: given a boolean formula in conjunctive normal form (a set of clauses) is it contradictory (or unsatisfiable) -- i.e., is it false for any truth assignment to all its variables?

These problems are actually identical, since one can transform the one into the other by taking the complement of the input and then applying the DeMorgan laws. The second is more natural in theorem proving. Given a set of axioms, we want to prove a theorem. We take the conjunction of all axioms and the negation of the theorem and hope to prove a contradiction.

A popular procedure to prove that a set of clauses is contradictory is the resolution procedure ([10]). It uses the resolution rule to generate new clauses. A proof of contradiction is obtained if the empty clause is derived. Its dual, which was actually known earlier [8], uses the consensus rule to generate new conjunctions. The input is a tautology iff the empty conjunction can be obtained. In this paper we investigate the resolution procedure for an input which consists of a set of clauses. All the results can be restated and will hold automatically for the dual problem. We consider various procedures which use the resolution principle and compare their computing powers. For some of them we will prove exponential lower bounds on their run-time.

The outline of the paper is as follows: In Section 2 we state some definitions and introduce our notation. In Section 3 we describe Tseitin's [13] ingenious construction of sets of clauses associated with graphs. We will use these sets throughout this paper. Tseitin used them to prove an exponential lower bound for regular resolution. Unfortunately his proof is informal, unintuitive and does not contain most of the details. In Section 6 we will give a simplified, hopefully intuitive, and detailed proof for Tseitin's main result. We also indicate in this section how to prove an exponential lower bound for the Davis Putnam procedure. Actually, it follows almost immediately from one of the lemmas. The more general result, the exponential lower bound for regular resolution, follows from a much more involved proof. In Section 4 we prove some general lemmas which will be used in the following sections. Although we have not been able to show that the unrestricted resolution procedure has an exponential lower bound, we believe that such lemmas as those in Section 4 might be used to prove this result. In Section 5 we introduce several versions of bounded resolution and iterated bounded resolution. They all use the resolution rule but allow generation of new clauses only if their size is not larger than some given bound. In the case of iterated bounded resolution the bound is increased whenever the current bound is found to be too small. We prove: (1) no proposed bounded resolution procedure always works correctly; and (2) all iterated bounded resolution procedures are exponential. Our attempts to find straightforward proofs for these results have failed. Our proofs make a nontrivial use of Tseitin's sets of clauses which correspond to graphs. In Section 7 we investigate what are the best results one might obtain using the graph examples. It turns out that we can almost achieve them by using graphs as suggested by Kirkpatrick [7]. Section 8 is devoted to extended resolution. We show that any proof tree using resolution can be transformed into another proof tree using extended resolution such that (1) the tree is regular, and (2) it contains only clauses of size at most three. The new tree will not use many more clauses than the original one. While (1) is not known to hold for resolution, (2) is shown not to hold for resolution in Section 5. We also include Tseitin's observation that extended resolution gives a polynomial proof for contradictory systems of clauses associated with graphs. The above does not hold for regular resolution and we conjecture that it also fails for unrestricted resolution but we were not able to prove this result. Section 9 contains a summary and a list of open problems.

2. DEFINITIONS AND NOTATIONS

We assume that there is an unbounded set $\{x_1, x_2, \ldots, y, \ldots\}$ of variables. A literal is either a variable or the complement of a variable, denoted x and $\bar{x}$ respectively (or x^1 and x^0). A clause is a finite set of literals $\{\ell_1, \ldots, \ell_k\}$ such that no variable appears more than once, and is denoted by $\ell_1 \vee \ell_2 \vee \ldots \vee \ell_k$. A formula (in conjunctive normal form (CNF)) is a finite set of clauses. A valuation (or assignment) is a function v from the set of literals to $\{0,1\}$ (zero is to be thought of as meaning false and one as meaning true). The value (relative to v) of the literal x $(\bar{x})$ is $v(x)$ $(1 - v(x))$. The value of the clause $\ell_1 \vee \ell_2 \vee \ldots \vee \ell_k$ is the maximum of the values of the ℓ_i's. (By convention if $k = 0$ we denote the clause by Λ, and define its value to be 0.) The value of a formula is 1 if the values of all its clauses are 1, and 0 otherwise.

A formula is satisfiable if there is some valuation for which the value of the formula is 1, and unsatisfiable (contradictory) otherwise. It is well known that any 0,1 valued function of k arguments may be represented by a formula in CNF with variables $\{x_1, \ldots, x_k\}$. Two formulas are equivalent iff they have the same value for all valuations. Cook [1] has observed that any formula in CNF of length n can be transformed into an equivalent formula of length O(n) in CNF with at most three literals per clause. We denote the latter by CN3.

In this paper we are interested in the following problem: given a formula in CNF (i.e. a set of clauses), is it contradictory?

Two clauses C_1 and C_2 are said to conflict if there are literals in C_1 which appear complemented in C_2. If C_1 and C_2 do not conflict then we denote by $C_1 \vee C_2$ the clause which contains all literals in C_1 and C_2. The clauses C_1 and C_2 are said to clash if there is exactly one literal in C_1 which appears complemented in C_2. If $C_1 \vee x$ and $C_2 \vee \bar{x}$ clash, then their resolvent is the clause $C_1 \vee C_2$. We will say that $C_1 \vee C_2$ is obtained from $C_1 \vee x$ and $C_2 \vee \bar{x}$ by applying the resolution rule, or by annihilating x.

A resolution proof of unsatisfiability of a set of clauses S is a sequence of clauses $C_1, \ldots, C_k$ such that $C_k = \Lambda$ and for $0 \leq i \leq k-1$, C_{i+1} is the resolvent of some pair from $S \cup \{C_1, \ldots, C_i\}$.

Robinson [10] showed that a set of clauses is unsatisfiable iff there is a resolution proof of its unsatisfiability.

A (resolution) proof tree for a clause C (using S) is a binary tree T whose vertices contain clauses in such a way that (1) C appears at the root of T; (2) every leaf of T contains one of the clauses in S, and (3) the clause in any vertex which is not a leaf is the resolvent of the clauses at its sons.

If the clauses C_1 and C_2 have proof trees T_1 and T_2, and can be resolved to get C, we define the corresponding operation on trees $T = \text{Res}(T_1, T_2)$. T is the proof tree for C whose two main subtrees are T_1 and T_2.

A resolution proof tree (using S) is a proof tree for the empty clause using S.

The complexity of a proof tree T, N(T), is the number of distinct clauses which appear on vertices of T.

Note that a proof tree T using S is just a way to represent a resolution proof of the unsatisfiability of S and that N(T) is actually the length of the resolution (straight line) proof defined above.

For a proof tree T using S and a vertex i in T let C_i be the clause at i and T_i the subtree with root i. Obviously T_i is a proof tree for C_i using S. If i is not a leaf we denote its sons by i_1 and i_2 and the variable annihilated at i by x_i. The height of T is the length of the longest path in T (i.e. between its root and one of the leaves).

We will consider several procedures which use the resolution rule. Most of them will be nondeterministic, i.e. contain some choices which are not specified. The complexity of a given procedure on an input of size n is the number of distinct clauses it generates. Note that this must be polynomially related to the run-time of the procedure. Considering procedures for proving unsatisfiability of a set of clauses, there are two kinds of results one would like to obtain. The first is to show that a given procedure has a polynomial complexity. This would imply P = NP if the procedure is deterministic and would imply that NP is closed under complementation if it is nondeterministic. So far none of the procedures has been shown to have a polynomial complexity and many researchers conjecture that none exists since they believe that P $\neq$ NP and NP is not closed under complementation.

The second is to show that known procedures have exponential complexity. By doing this to a given procedure, one excludes it from being a candidate for the first kind of results. Although this does not imply anything about the open problems mentioned above, one hopes that obtaining exponential lower bounds for the run-time of specific problems (which is in some cases not an easy task) will require the development of techniques which will help to analyze the run-time of procedures in general.

We now describe the different procedures which we consider in this paper.

1. The (unrestricted) resolution procedure is simply constructing a resolution proof tree using the given set of clauses S.

Let T be a proof tree for some clause C. T is regular if there is no vertex i in T such that C_i contains some literal which is annihilated in T_i.

Note that a proof tree T is regular iff no path from a leaf to the root of T contains the annihilation of any variable more than once.

2. Regular resolution is simply constructing a regular resolution proof tree using S.

One should note that a set of clauses is unsatisfiable iff it has a regular proof tree. The "if" part is obvious. The "only if" part holds because the inductive proof of the completeness of resolution happens to construct a regular tree.

3. Extended resolution is applying the resolution procedure to a larger set of clauses $S' \supseteq S$ which is obtained from S by using the extension rule described in Section 8. It is valid to consider S' since by its construction S' is satisfiable iff S is satisfiable.

4. Several bounded resolution procedures are defined in Section 5. They all use the resolution rule in a restricted manner, allowing the generation of a new clause only if its size is not larger than some given bound. The iterated bounded resolution procedures are obtained by iterating the corresponding bounded resolution procedure with an increasing bound.

5. The Davis Putnam procedure (DPP) [4] is the following routine:

I. Choose a variable x.

II. Replace all the clauses which contain x (or $\bar{x}$) by the set of all clauses which can be obtained from two of them by the resolution tule.

III. (a) If the empty clause is present, then the original set is unsatisfiable.

(b) If the current set is empty, then the original set is satisfiable.

(c) Otherwise, repeat steps I - III for the new set of clauses.

Step II is referred to as eliminating x.

The procedure above works since one can show that all the transformations used preserve satisfiability (unsatisfiability).

Examples have been found for sets of clauses for which the DPP has exponential complexity ([2],[11]). However, those exploited redundancy in the formula, which can easily be removed by adding the subsumption rule: delete from the current set of clauses S any clause C such that there is another clause C' in S, with $C' \subseteq C$ (C' subsumes C).

From now on we will assume DPP includes the subsumption rule (between steps II and III). We call an execution of steps I - III a stage. Note that if the original set is unsatisfiable one can construct a proof tree by tracing back the history of the empty clause (as was done for resolution proof). We call such a tree a DP-tree. Note that every DP tree T satisfies the following property: x is annihilated above a point at which y is annihilated in T iff y was eliminated before x. Hence every DP tree is regular but not every regular tree is a DP-tree.

We will use extensively sets of clauses corresponding to graphs. We present below the notation which we use for dealing with graphs.

All the graphs will be connected, undirected, and without self loops. A graph is a pair $\bar{G} = (\bar{V},\bar{E})$, where $\bar{V}$ is its set of vertices $(u,v,w\ldots)$ and $\bar{E} \subseteq \bar{V} \times \bar{V}$ its set of edges. A graph $G = (V,E)$ is a subgraph of $\bar{G}$ if $V \subseteq \bar{V}$ and $E \subseteq \bar{E}$ (in the literature G is sometimes called a partial subgraph of $\bar{G}$). We will use the following operations on subgraphs of $\bar{G}$:

$G = G_1 \cup G_2$, where $V = V_1 \cup V_2$ and $E = E_1 \cup E_2 \cup E_{12}$.

E_{12} is the set of edges between vertices in V_1 and V_2.

$G = G_1 \cap G_2$ if $V = V_1 \cap V_2$ and $E = E_1 \cap E_2$

$G = G_1 - G_2$ if $V = V_1 - V_2$ and E consists of all edges in E_1 between two vertices in V.

The graph is of degree ℓ if every vertex is incident with at most ℓ edges.

3. SETS OF CLAUSES CORRESPONDING TO GRAPHS

Usually when someone wants to prove something about a proposed procedure for theorem proving, he faces the following problem: How does one construct a nontrivial infinite collection of unsatisfiable formulas? One set of examples is the following:

Let $\beta_{n,k}$ be a formula in CNF over n variables which contains all clauses of size $k \leq \lceil n/2 \rceil$ † such that the variables in any term are either all complemented or all noncomplemented. For every $n \geq 0$ and $k \leq \lceil n/2 \rceil$, $\beta_{n,k}$ is unsatisfiable since every assignment has either k zeros or k ones, and thus at least one of the clauses must

† $\lceil n/2 \rceil$ is the smallest integer greater than or equal to n/2.

have the value 0.

The previous example is derived from the following general prin ciple: choose a set of clauses which altogether exclude the 2^n possible assignments (which can be viewed as the corners of an n-dimensional cube).

Another general principle is: take an invalid statement and formalize its instances in the propositional calculus. As an exampl consider the "occupancy problem" by Cook and Karp (c.f. [5] p. 45) The statement is, "n objects can occupy all of n+1 holes".

Another example: for every n one can formalize the wrong state ment: "A clique of size n can be colored with less than n colors".

Tseitin used the following invalid statement

A: "If you compute the sum modulo 2 of a number of binary variable in such a way that every variable appears exactly twice you ge

To construct a formula α in CNF which expresses this statement we take a graph and label its vertices with zeros and ones and its edges with literals. Next, for every vertex v we express in CNF th statement

B: "The sum modulo 2 of all the edge labels incident with v is equal to the vertex label of v."

α is the union of all these sets of clauses. Summing up statement for all the vertices implies that the sum modulo 2 of all vertex labels must be 0, since each edge label appears in the sum exactly twice (once for each of its endpoints). By choosing a labelling with sum modulo 2 of vertex labels equal 1, we formalize statement in an indirect manner. By choosing a graph with a small degree we can express statement B by a relatively short formula. This is essential, since otherwise we will have too long an input for whic even a brute force algorithm (like truth table computation) takes non-exponential run-time.

Formally, let $G = (V,E)$ be a given undirected graph without loops (i.e. there is no edge (v,v)). Let ℓ be a labelling which assigns to every vertex v a label $\varepsilon_v \in \{0,1\}$, the parity of v, and each edge a literal in such a way that no two edges are labelled by a literal corresponding to the same variable.

Consider a vertex v in V. Assume that v has degree k and $z_1, z_2, \ldots, z_k$ are the literals which label the edges incident with v. A clause C corresponds to the vertex v iff

$$C = z_1^{\varepsilon_1} \vee \ldots \vee z_k^{\varepsilon_k}$$

and the number of complemented z_i's is of parity opposite to the parity of v, i.e.

$$\sum_{i=1}^{k} (1 - \varepsilon_i) \neq \varepsilon_v$$

which will be referred to as the parity condition with respect to v.†

Note that there are exactly 2^{k-1} distinct clauses which correspond to v. The following lemma proves that the formula which consists of all of them is equivalent to statement B mentioned before, namely that the parity of the number of z_i's assigned the value 1 must be ε_v. Intuitively it is clear, since each clause which corresponds to v excludes one assignment which does not satisfy the above, and all the clauses exclude all unwanted assignments.

Lemma 3.1: The assignment δ given by $z_1 = \delta_1, \ldots, z_k = \delta_k$ satisfies all the clauses corresponding to v iff

$$\sum_{i=1}^{k} \delta_i = \varepsilon_v.$$

---

† $\sum_{i=1}^{k} \delta_i = \delta_1 + \delta_2 + \ldots + \delta_k$ is the sum modulo 2 of $\delta_1, \delta_2, \ldots, \delta_k$.

Proof: If $\sum_{i=1}^{k} \delta_i \neq \varepsilon_v$, then $C = z_1^{1-\delta_1} \vee \ldots \vee z_k^{1-\delta_k}$ corresponds to v and is not satisfied by the assignment δ.

Assume $\sum_{i=1}^{k} \delta_i = \varepsilon_v$ and let $C = z_1^{\varepsilon_1} \vee \ldots \vee z_k^{\varepsilon_k}$ be an arbitrary clause corresponding to v. The only assignment which does not satisfy C is $z_1 = 1 - \varepsilon_1, \ldots, z_k = 1 - \varepsilon_k$. Since $\sum_{i=1}^{k} (1 - \varepsilon_i) \neq \varepsilon_v$ it must differ from the assignment δ. Thus δ must satisfy C. ∎

We denote by $\alpha(G,\ell)$ the set of all clauses which correspond to all vertices in G (ℓ is the labelling).

Claim: $\alpha(G,\ell)$ is unchanged if we apply the following transformations to the labelling:

Transformation 1: change exactly one edge label (from z to $\bar{z}$) and simultaneously change the vertex labels of its endpoints.

Transformation 2: change all edge labels along a path from vertex u to vertex v and simultaneously change ε_u and ε_v.

Transformation 2 is a sequence of applications of transformation The latter does not affect $\alpha(G,\ell)$ since if a clause C corresponds to a vertex w under the old labelling it does so under the new labelling for $w \notin \{u,v\}$ this is obvious since nothing has changed; for w = u (w = v) the change of ε_u (ε_v) and of z to $\bar{z}$ implies a double change in parity. Hence C satisfies the parity condition with respect to u (v) under the new labelling.

Let $\varepsilon(G,\ell) = \sum_{v \in V} \varepsilon_v$.

Lemma 3.2: For a connected graph G and a labelling ℓ, $\alpha(G,\ell)$ is satisfiable iff $\varepsilon(G,\ell) = 0$.

Proof: Assume $\alpha(G,\ell)$ is satisfiable. Using Lemma 3.1 and summing (mod 2) all ε_v's we get $\varepsilon(G,\ell) = 0$, since every δ_i appears twice (once for each endpoint of the edge labelled by z_i.

Assume $\varepsilon(G,\ell) = 0$. Using transformation 2 we can set all vertex labels to 0. If the new edge labels are $z'_1,\ldots,z'_n$ the assignment $z'_i = 0$ for $1 \leq i \leq n$ satisfies $\alpha(G,\ell)$. ∎

Thus if $\varepsilon(G,\ell) = 1$, $\alpha(G,\ell)$ is unsatisfiable, since it formalizes statement A mentioned before. The following lemma proves that every proof tree using $\alpha(G,\ell)$ must use at least one clause corresponding to each vertex in G.

Lemma 3.3: Assume $\alpha(G,\ell)$ is contradictory and T is a tree which proves the contradiction. Let $V_1 = \{v|$ a clause corresponding to v appears on a leaf of T$\}$, then $V_1 = V$.

Proof: Assume the claim is wrong and that $v \in V - V_1$. Define a new labelling ℓ' by changing ε_v. By Lemma 3.2, $\alpha(G,\ell)$ is contradictory implies $\varepsilon(G,\ell) = 1 \Rightarrow \varepsilon(G,\ell') = 0 \Rightarrow \alpha(G,\ell')$ is satisfiable. But all the clauses on the leaves of T belong also to $\alpha(G,\ell')$ since none correspond to v. Thus we have a proof tree using a satisfiable set of clauses -- a contradiction which proves the lemma. ∎

From now on we consider a fixed connected graph $\bar{G} = (\bar{V},\bar{E})$ with a fixed labelling ℓ such that $\varepsilon(\bar{G},\ell) = 1$. Thus $\alpha(\bar{G},\ell)$ is unsatisfiable. We will refer to $\alpha(\bar{G},\ell)$ as the original set of clauses. We will identify the edges and their labellings if no confusion will arise.

4. SOME GENERAL LEMMAS

In this section we prove three general lemmas:

1. The Correspondence Lemma, which states the correspondence between operations on a graph G and a regular derivation using the clauses corresponding to G.

2. The Regularity Lemma, which states that if we can prove a given clause C by a resolution tree, then we can prove a clause C', $C' \subseteq C$, by a regular tree.

3. The Substitution Lemma, which proves the validity of substituting a given tree for a subtree in another given proof tree.

These three lemmas will be used in the sequel for proving most of the theorems. We still hope that by using these lemmas one will be able to prove an exponential lower bound for the general resolution procedure.

Let $G = (V,E)$ be a connected subgraph of $\bar{G}$. <u>The boundary of G</u>, F (or F_G when G is not clear from the context) is the set of edges of $\bar{G}$ which are incident with at least one vertex of G and do not belong to E. The <u>exterior boundary</u> (<u>interior boundary</u>) of G, F^{ext} (F^{int}), consists of edges in the boundary of G which are incident with exactly one vertex (two vertices) of G.

A clause C is <u>associated with G</u> if its variables are the labels of the edges in the boundary of G.

If C is associated with G, then the <u>parity of C</u>, $\delta(C)$, is the parity of the number of complemented variables in C which correspond to the exterior boundary of G, i.e. if $C = z_1^{\varepsilon_1} V \ldots V z_k^{\varepsilon_k}$ then

$$\delta(C) = \sum_{z_i \in F^{ext}} (1 - \varepsilon_i) .$$

The <u>parity of G</u>, $\varepsilon(G)$, is defined to be the sum modulo 2 of the vertex labels of G. C is a <u>regular clause associated with G</u> if C is associated with G and the parity of C is opposite the parity of G, i.e. $\delta(C) \neq \varepsilon(G)$.

C is a regular clause if there is a connected subgraph G of $\bar{G}$ such that C is a regular clause associated with G.

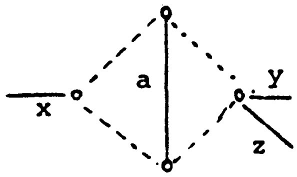


Figure 3.1

A subgraph G of $\bar{G}$ and all edges of $\bar{G}$ which are incident with its vertices.

Example: Consider the subgraph $G = (V,E)$ of $\bar{G}$ in Figure 3.1. V consists of the circles and E consists of the dotted lines. Assume that all vertex labels are 0, thus $\varepsilon(G) = 0$, $F^{int} = \{a\}$ and $F^{ext} = \{x,y,z\}$. Let $C_1 = a \vee x \vee y \vee z$, $C_2 = \bar{a} \vee x \vee y \vee z$ and $C_3 = a \vee x \vee y \vee \bar{z}$. All three clauses are associated with G, but only C_3 is a regular clause since $\delta(C_3) = 1$ while $\delta(C_1) = \delta(C_2) = 0$.

One can make the following observations:

i) for every clause C, there can be at most one connected subgraph G of $\bar{G}$, such that C is a regular clause associated with G.

Proof: Assume C is a regular clause associated with G. By deleting all edges of the boundary of G from $\bar{G}$ we obtain a possibly unconnected graph G', where G is one of its connected components. If the interior boundary of G is nonempty, C can be associated only with G. This is also the case when G' has more than two connected components. Otherwise, i.e. in case $F^{int} = \emptyset$ and $G \cup G_1 = \bar{G}$. (G_1 is the other component), C is also associated with G_1. C cannot be a regular clause associated with G_1 since $\varepsilon(G) + \varepsilon(G_1) = \varepsilon(\bar{G}) = 1 \Rightarrow \varepsilon(G_1) \neq \varepsilon(G) \neq \delta(C) \Rightarrow \varepsilon(G_1) = \delta(C)$.

The second observation is obvious:

ii) if C is one of the original clauses which corresponds to a vertex v in $\bar{G}$, then C is a regular clause associated with the subgraph $G = (v,\emptyset)$ consisting of an isolated vertex.[†]

The next lemma proves that if T is a regular proof tree for a clause C, then C is a regular clause. The lemma specifies exactly the connected subgraph G with which C is associated.

Lemma 4.1 (The Correspondence Lemma): If T is a regular proof tree for a clause C, then C is a regular clause associated with some connected subgraph $G = (V,E)$ of $\bar{G}$ where

$V = \{v \mid v \in \bar{V}$ and a clause corresponding to v appears on a leaf of T$\}$

and

$E = \{x \mid x$ is annihilated in T$\}$.

Proof: Induction on the height h of T. The basis of the induction ($h = 0$) follows immediately from observation (ii) preceding the lemma.

Induction step: Let T be the derivation in Figure 3.2 and assume $x \vee C_1$ ($\bar{x} \vee C_2$) is a regular clause associated with G_1 (G_2) which satisfies the hypothesis of the lemma.

T:

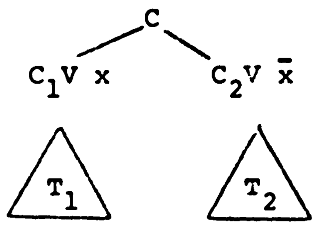


Figure 3.2

Without loss of generality there are four cases:

† We identify the vertex v and the singleton {v}.

1. $G_1 = G_2$ (the subgraphs are identical).
2. $V_1 \cap V_2 = \emptyset$ † (they are vertex disjoint).
3. $V_1 = V_2$ and there is $z \in E_1 - E_2$
4. $V_1 \cap V_2 \neq \emptyset$ and there is $u \in V_1 - V_2$.

Claim: Cases 3 and 4 are not possible. We show that in both cases $E_1 \cap F_2 \neq \emptyset$, i.e. there is a variable z annihilated in T_1 such that either z or $\bar{z}$ appears in $C_2 V \bar{x}$ -- a contradiction since T is regular. For case 3 it is obvious that $z \in E_1 \cap F_2$. For case 4 let $v \in V_1 \cap V_2$. G_1 is connected, thus there is a path in G_1 from u to v. There must be an edge z on this path such that one of its endpoints is in V_2 and one is not. Obviously $z \in E_1 \cap F_2$. This completes the proof of the claim.

To complete the proof of the lemma we now consider cases 1 and 2.

Case 1. $G_1 = G_2$ and thus $F_1 = F_2$. $x V C_1$ and $\bar{x} V C_1$ consist of the same variables and have exactly one conflict (x vs. $\bar{x}$), thus $C_1 = C_2 = C$. Since the parity of $C_1 V x$ and $C_2 V \bar{x}$ both differ from the parity of $G_1 = G_2$, they must be the same. Hence $x \in F_1^{int} = F_2^{int}$. Let $G = (V_1, E_1 \cup x)$. It is easy to check that the pair (C,G) satisfies the induction hypothesis. The parity of G differs from the parity of C since $\varepsilon(G) = \varepsilon(G_1)$ and $\delta(C) = \delta(C_1 V x)$.

Case 2. $V_1 \cap V_2 = \emptyset$. Let $G = (V_1 \cup V_2, E_1 \cup E_2 \cup x)$. Obviously, G is connected, C is associated with G, $E = E_1 \cup E_2 \cup x$ consists exactly of the variables annihilated in T and $V = V_1 \cup V_2$ consists exactly of those vertices which correspond to the clauses on the leaves of T. To complete the proof that the pair (C,G) satisfies the induction hypothesis we now show that G has parity opposite to that of C.

† for i = 1,2 $G_i = (V_i, E_i)$, F_i, F_i^{ext}, F_i^{int} are defined in the obvious way.

Since $\delta(C_1 V\ x) \neq \varepsilon(G_1)$ and $\delta(C_2 V\ \bar{x}) \neq \varepsilon(G_2)$,

$$\delta(C_1 V\ x) + \delta(C_2 V\ \bar{x}) = \varepsilon(G_1) + \varepsilon(G_2) = \varepsilon(G).$$

We show below that

(1) $\delta(C_1 V\ x) + \delta(C_2 V\ \bar{x}) \neq \delta(C)$

which will imply that $\delta(C) \neq \varepsilon(G)$ and will complete the proof of the lemma.

To prove (1), recall that if $C = z_1^{\varepsilon_1} V \ldots V z_r^{\varepsilon_r}$, then

$$\delta(C) = \sum_{z_i \in F^{ext}} (1 - \varepsilon_i)$$

But $F_1^{ext} \cap F_2^{ext} \equiv x \cup F_{12}$ and $F_{12} \subseteq F^{int} - x$ (See Figure 3.3). In $\delta(C_1 V\ x) + \delta(C_2 V\ \bar{x})$ the terms corresponding to F^{ext} appear exactly once, the terms for F_{12} appear exactly twice (hence they can be dropped) and since x appears once as x and once as $\bar{x}$,

$$\delta(C_1 V\ x) + \delta(C_2 V\ \bar{x}) = \delta(C) + 0 + 1$$

and (1) follows.

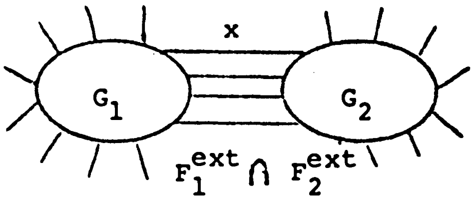


Figure 3.3

<u>Corollary 4.1</u>: Let T be the regular proof tree of Figure 3.2 and let C ($C_1 V\ x$; $C_2 V\ \bar{x}$) be associated with G = (V,E) (G_1, G_2 respectively). Then $x \in E$ and if we delete x from E and obtain G' = (V,E − x) there are two cases:

Case 1: G' is still connected. In this case $G' = G_1 = G_2$ and we will say that <u>x breaks a cycle in G</u>.

Case 2: G' is not connected. In this case G_1 and G_2 are its connected components and we will say <u>x splits G</u> (into G_1 and G_2).

Consider the following procedure. It generates a tree such that every vertex i corresponds to some connected subgraph of $\bar{G}$.

```
procedure deletion process (Ḡ);
    begin
        procedure  buildtree (G,i);
            begin
                if G is a single vertex then return;
                generate two new vertices j+1, j+2 as sons of i;
                choose x ∈ E;
                if x splits G then let G_{j+1}, G_{j+2} be the connected
                    components of G' else let G_{j+1}, G_{j+2} ← G';
                j ← j+2;
                buildtree (G_{j+1}, j+1);
                buildtree (G_{j+2}, j+2);
            end buildtree ;
        j ← 1;
        G_1 ← Ḡ;
        buildtree (Ḡ,1)
    end deletion process;
```

We use the term deletion process for executing the procedure above for some specified sequence of choices for the deleted edge x. By Corollary 4.1 every regular proof tree has a corresponding deletion process: it always chooses the edge whose label is annihilated at the current vertex. But one can also prove

Lemma 4.2: For every deletion process for $\bar{G}$ there is a corresponding regular proof tree which uses the clauses which correspond to $\bar{G}$.

The proof is by induction on the size of $\bar{E}$, and is omitted.

Thus the correspondence between regular proof trees and deletion processes is one to one.

The next two lemmas apply to all resolution trees. They will be used only in the next section where we deal with bounded resolution, which also allows nonregular resolution trees.

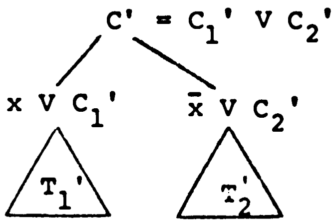
We now show that every clause in every proof tree contains a regular clause.

<u>Lemma 4.3 (The Regularity Lemma)</u>: Let T' be a proof tree for the clause C'. Then there is a $C \subseteq C'$ and a regular proof tree T for C, such that $\alpha \subseteq \alpha'$ where α (α') is the set of clauses on the leaves of T (T').

<u>Proof</u>: Induction on the height of T'.

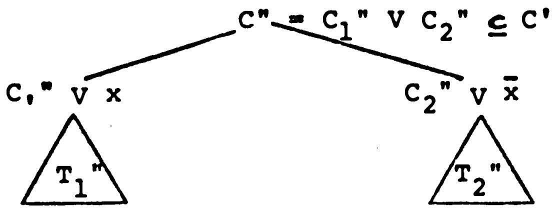
Basis - trivial

Induction step: Let T' be the tree



Apply the induction hypothesis to T_1' (T_2') to obtain a regular tree T_1'' (T_2'') which proves $\hat{C}_1$ ($\hat{C}_2$) with $\hat{C}_1 \subseteq x \vee C_1'$ ($\hat{C}_2 \subseteq \bar{x} \vee C_2'$).

If $x \notin \hat{C}_1$ ($\bar{x} \notin \hat{C}_2$) then $T = T_1''$ ($T = T_2''$) and $C = \hat{C}_1$ ($C = \hat{C}_2$) satisfy the hypothesis of the lemma. Otherwise let $\hat{C}_1 = C_1'' \vee x$ and $\hat{C}_2 = C_2'' \vee \bar{x}$ and let T" be the tree



T" needs not be regular because a literal which appears in $C_1" \vee x$ ($C_2" \vee \bar{x}$) might be annihilated in $T_2"$ ($T_1"$). So let

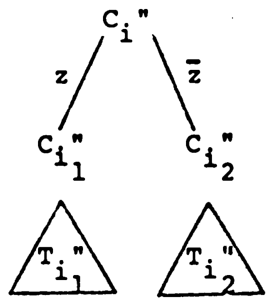
$$P = \{z \mid z \in C_1",\ z \text{ is annihilated in } T_2"\} \cup \{z \mid z \in C_2",\ z \text{ is annihilated in } T_1"\}.$$

If $P = \emptyset$ then T" is regular and T = T" and C = C" will do.

Otherwise we eliminate in T" all annihilations of z's in P by deleting corresponding subtrees. T" decomposes into a collection of separate subtrees and we construct from them a new tree T by performing all possible annihilations.

More precisely we construct T by the following inductive process:

If $T_i"$ consists of a single vertex, then $T_i = T_i"$. If $T_i"$ is the tree



then

(1) If $z \in P$ ($\bar{z} \in P$), then $T_i = T_{i_1}$ ($T_i = T_{i_2}$), i.e. the other subtree is deleted to prevent annihilation; otherwise

(2) If $z \notin C_{i_1}$ ($\bar{z} \notin C_{i_2}$), then $T_i = T_{i_1}$ ($T_i = T_{i_2}$) i.e. the other subtree is deleted since we do not need to annihilate z in the new tree); otherwise

(3) $T_i \leftarrow \mathrm{Res}(T_{i_1}, T_{i_2})$.

Let $T = T_1$ (where 1 is the root) and $C = C_1$. For i in T, an easy induction on the height of T_i proves that

(1) T_i is regular

(2) $C_i \subseteq C_i'' \vee P$ (recall that a clause is a set of literals).

Thus T is regular and $C \subseteq C'' \subseteq C'$, since $P \subseteq C''$. Hence T and C satisfy the required properties.

In all cases $\alpha \subseteq \alpha'$ since T is obtained from T' by using only subtrees of T'. This completes the proof.

We will need an operation called <u>substitution</u> which we now define.

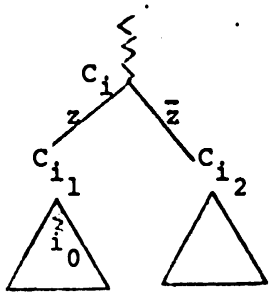
Let T be a proof tree and i_0 a vertex in it, and assume we are given another tree T'_{i_0} with $C'_{i_0} \subseteq C_{i_0}$ where C'_{i_0} is the clause at the root of T'_{i_0}.

We describe below what we mean by the statement "T' is obtained from T by substituting T'_{i_0} for T_{i_0} in T."

Informally we go along the path from i_0 to the root of $T - 1$. We start with T'_{i_0} and perform all possible resolutions with subtrees which are incident with this path.

More precisely we construct T' by the following inductive process

T'_{i_0} is the given tree.



Assume i_1 is on the path, i is its father, i_2 is its brother and z is annihilated at i. Assume that T'_{i_1} has been defined. Then

$$T'_i = \begin{cases} \text{if } z \notin C'_{i_1} \text{ then } T'_{i_1} \\ \text{else } \mathrm{Res}(T'_{i_1}, T_{i_2}) \end{cases}$$

We define $T' = T'_1$.

Lemma 4.4 (The Substitution Lemma): Assume T is a proof tree, T'_{i_0} is a proof tree for $C'_{i_0} \subseteq C_{i_0}$ and T' is obtained from T by substituting T'_{i_0} for T_{i_0} in T. Then T' is a proof tree.

Proof: For all i on the path from i_0 to the root, $C_i' \subseteq C_i$, where C_i' is the clause at the root of T_i' constructed above. This is proved by induction on the distance of i from i_0. The claim of the lemma follows from the case $i = 1$. ∎

We now specify $\bar{G}$ to be the $k \times k$ square grid in Figure 3.4

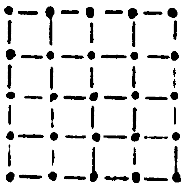


Figure 3.4: 5 × 5 square grid

By the definition of the set of clauses corresponding to $\bar{G}$, $\alpha_n \equiv \alpha(\bar{G},\ell)$ is a formula in CN4, over $n = 2k(k-1)$ variables and thus it contains $O(n)$ clauses. We now prove Lemma 4.5 which will be needed in the sequel.

Let T be a regular proof tree and let i be a vertex on the tree. Let $G_i = (V_i,E_i)$ be the subgraph associated with C_i by the Correspondence Lemma. Recall that V_i is the set of vertices in $\bar{G}$ such that clauses corresponding to them appear on leaves of T_i. For $V \subseteq \bar{V}$ let $\#(V)$ be the number of vertices in V which appear on the perimeter of the square.

Lemma 4.5: There is an i_0 such that $k < \#(V_{i_0}) \leq 2k$. Also $|F_{i_0}^{ext}| > k$.

Proof: By Lemma 3.3, $\#(V_1) = 4k - 2$ (where 1 is the root) since $V_1 = V$. For a leaf j, $\#(V_j) \leq 1$, since $|V_j| = 1$. If i_1 and i_2 are the sons of i and $\#(V_{i_1}) \geq \#(V_{i_2})$, then $\#(V_{i_1}) \geq \frac{1}{2}\#(V_i)$ ($V_i = V_{i_1} \cup V_{i_2}$).

Thus by going along a path from the root 1 to a leaf, choosing always i_1, the value $\#(V_i)$ decreases from $4k - 2$ to 1 and never decreases by more than half in any one step. Thus the existence of i_0 is guaranteed.

To check that $|F_i^{ext}| \geq k$ one has to consider several cases. In each case G_{i_0} has vertices in common with different edges of the square. We leave the verification of this fact to the interested reader. ∎

Note that the first half of the lemma holds also for general (not necessarily regular) proof trees, where V_i has the same interpretation.

5. ON THE VALIDITY AND COMPLEXITY OF BOUNDED RESOLUTION

In this section we consider certain procedures based on (not necessarily regular) resolution for checking whether a formula in CN3 is unsatisfiable. We get the following results:

1) All of the proposed procedures which are forced to run in polynomial time do not always work -- i.e., they do not identify all unsatisfiable formulas.

2) Those which always work must run in exponential time. The exponential lower bounds for these procedures do not follow from Tseitin's lower bound for regular resolution since these procedures also allow nonregular resolution trees.

We consider three versions of bounded resolution (BR): we assume that there is an integer k such that the size of each clause generated is bounded by k. All three versions use an operation RESOLVE(k) which adds to the current set of clauses α all those clauses of size $\leq k$ obtained from two of the clauses in α by the resolution rule. Two of the versions use the operation EXPAND(k) which replaces each clause C of size less than k by all terms $\hat{C}$ of size k such that $C \subseteq \hat{C}$. Let ALL(k) be a predicate which is true iff all terms of size k are present. The three routines are:†

```
BR1(k): do
          EXPAND(k)
          RESOLVE(k)
        end
        repeat until no new clauses are generated;
     1
        if ALL(k) then output "unsatisfiable"
                  else output "satisfiable"

BR2(k): do
          RESOLVE(k)
        end
        repeat until no new clauses are generated;
      2
        EXPAND(k);
      3
        if ALL(k) then output "unsatisfiable"
                  else output "satisfiable"
```

† The numbers 1,2,3 denote positions in the programs. We will refer to them in the sequel.

```
BR3(k): do
          RESOLVE(k)
        end
        repeat until no new clauses are generated ;
        if the empty clause is present then output "unsatisfiable"
                                       else output "satisfiable"
```

These procedures generate $O(m^k)$ distinct clauses, where m is the number of variables. Hence they all run in polynomial time. For each of these procedures we would like to know:

(1) Does the procedure work correctly for some constant integer k? and if not,

(2) Does it work correctly for $k = k(n)$, some slowly growing function of n, where n is the size of the input? (Obviously $k = m$ will do.)

A positive answer to (1) would imply P = NP. A positive answer to (2) would yield a subexponential algorithm for the tautology problem (and hence to all P-complete problems) provided $k(n)$ is asymptotically slower than n^{ε} for every ε.

That question (1) has a negative answer seems obvious, but the simplest proof we could obtain for this result is distinctly non-trivial!

If k is not large enough, these procedures might give a wrong answer -- they might decide that the input is satisfiable while it actually is unsatisfiable. The possibility of such an error can be detected. Obviously if upon termination of BR1 (BR2, BR3) all terms of size k are present (the empty clause is present) the input is unsatisfiable. Otherwise, if a new term can be generated (by the resolution rule) which is larger than k and does not contain any of the present clauses one cannot conclude that the input is satisfiable, but rather that the bound k is too small. This leads to the <u>iterated bounded resolution</u> (IBR): IBR1 (IBR2, IBR3) is simply iterating BR1(k) (BR2(k), BR3(k)) with increasing k if the case mentioned above occurs.

These iterated bounded resolution procedures always give the correct answer. The question about them is:

(3) What is the time bound for the iterated bounded resolution procedures?

Note that we restricted the inputs to formulas in CN3; otherwise questions (1), (2) and (3) are meaningless.

<u>Example</u>: Recall $\beta_{n,k}$ defined in Section 3 and consider

$$\beta_{5,3} = \{(x_1 \vee x_2 \vee x_3), (x_1 \vee x_2 \vee x_4), \ldots, (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3), \ldots, (\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5)\}.$$

One can see immediately that BR(3) would not work and that BR(4) will work (where BR is any one of the versions).

Our attempts to answer questions (1) - (3) by first considering simple examples have failed. The seemingly simple question, "Is the bound k = 4 sufficient for m = 6?" (where m is the number of variables) was answered by J. Simon [12]. However, the proof is complicated and does not generalize. Obviously one can use the idea in the example above and consider $\beta_{m,\ell}$, $\ell \leq \lceil m/2 \rceil$. Obviously a bound $k \geq 2\ell-2$ is necessary for BR(k) to work for $\beta_{m,\ell}$. However, after using the usual trick of converting $\beta_{m,\ell}$ to CN3 it is no longer clear that a smaller bound would not suffice. Even if this method worked it would have settled question (1), but not questions (2) and (3), since the input is of length $n \approx m^{\ell}$ and the number of distinct clauses generated is $< n^2$ since one can show that the bound $k = 2\ell-2$ suffices.

In Theorem 5.1 we prove the equivalence of BR1, BR2 and BR3. This will enable us to speak about BR which will stand for each one of them. In Theorem 5.2 we will show that any proof tree for the set of clauses corresponding to the $k \times k$ square grid must generate clauses of size k. This will answer question (1) in the negative for BR3. However, a few observations will show that it actually settles all three questions for all three versions of (iterated) bou: resolution.

Before answering these questions we prove:

<u>Theorem 5.1</u>: For a fixed k all proposed bounded resolution versions are equivalent, i.e. they give the same answer to any given input.

<u>Proof</u>: First we show that BR1 and BR2 are equivalent. Let $term_1$, $term_2$ and $term_3$ be the clauses present at points 1, 2 and 3 in BR1 a BR2. We show that (1) for every $\hat{C}$ in $term_1$ there is $C \in term_2$ such that $C \subseteq \hat{C}$; and (2) for every C in $term_2$, every $\hat{C}$ of size k such tha $C \subseteq \hat{C}$ is in $term_1$. This will imply (by definition of EXPAND(k)) tha $term_1 = term_3$ and will complete the proof.

(1) Proof by induction on the level of resolution in BR1. For level 0 this follows from the definition of EXPAND(k). Assume $\hat{C} \in term_1$. There must be $\hat{\hat{C}} \subseteq \hat{C}$ which was obtained from $\hat{C}_1$ and $\hat{C}_2$:

$$\begin{array}{ccc} & \hat{\hat{C}} & \\ {}^{x}\diagup & & \diagdown^{\bar{x}} \\ \hat{C}_1 & & \hat{C}_2 \end{array}$$

By induction hypothesis there are $C_1, C_2 \in term_2$, $C_1 \subseteq \hat{C}_1$ and $C_2 \subseteq \hat{C}$ If $x \notin C_1$ ($\bar{x} \notin C_2$) then $C_1 \in term_2$ ($C_2 \in term_2$) and $C_1 \subseteq \hat{C}$ ($C_2 \subseteq \hat{C}$). Otherwise, BR2 generates C, the resolvent of C_1 and C_2. $C \in term_2$; $C \subseteq \hat{C}$.

(2) Proof by induction on the level of resolution in BR2. For level 0 it follows again by definition of EXPAND(k). Assume $C \in term_2$ and is obtained from C_1 and C_2:

$$\begin{tikzcd}[column sep=small] & C \arrow[dl, dash, "x"'] \arrow[dr, dash, "\bar{x}"] & \\ C_1 & & C_2 \end{tikzcd}$$

Let $\hat{C} \supseteq C$ be of size k. We show that $\hat{C} \in term_1$. Note that $|C_1 \cap C_2| < k$ ($x \in C_1$, $\bar{x} \in C_2$). There are two possibilities:

(a) $|C_1 \cap C_2| = k-1$. Thus $|C_1| = |C_2| = k$ and by the induction hypothesis they belong to $term_1$. Thus C will be generated in BR1 by RESOLVE(k) and $\hat{C}$ will be generated by the immediately following EXPAND(k).

(b) $|C_1 \cap C_2| \leq k-2$. Thus $(k - |C_1|) + (k - |C_2|) = 2k - |C_1| - |C_2| = 2k - |C_1| - |C_1 \cap C_2| - |C_2 - C_1| \geq k+2 - |C_1| - |C_2 - C_1| = k+2 - |C_1 \cup C_2| = k - |C|$. Following the inequality proved above, since $|\hat{C} - C| = k - |C|$ we can write $\hat{C} - C = D_1 \cup D_2$ such that $|D_i| \leq k - |C_i|$ for $i = 1,2$. Thus $\hat{\hat{C}}_i = C_i \cup D_i$ satisfies $|\hat{\hat{C}}_i| \leq k$. Let $\hat{C}_i$ be a clause of size k obtained from $\hat{\hat{C}}_i$ by adding to it enough literals from $\hat{C} - \hat{\hat{C}}_i$. By induction hypothesis, $\hat{C}_1$ and $\hat{C}_2$ belong to $term_1$. So C, which is obtained from them by the resolution rule, belongs to $term_1$.

Now we show that BR2 and BR3 are equivalent. We prove that $\Lambda \in term_2$ iff ALL(k) is true for $term_3$. Since up to point 2 BR2 and BR3 are the same, this completes the proof.

If $\Lambda \in term_2$, then ALL(k) is true for $term_3$ by definition of EXPAND(k). To prove the converse we need the following definition: A set of terms satisfies ALL'(ℓ) if, for every clause $\hat{C}$ of size ℓ, there is a term C in the set such that $C \subseteq \hat{C}$. If ALL(k) is true for $term_3$, then ALL'(k) is true for $term_2$. We show below that if ALL'(i), $i > 0$, is true for a set of clauses, then ALL'(i-1) is true for the set obtained by applying RESOLVE(k). But RESOLVE(k) does not change $term_2$. Thus ALL'(0) must be true for $term_2$, i.e. $\Lambda \in term_2$. Let α be a set of clauses which satisfies ALL'(i), $i > 0$, and let β be the set obtained by applying RESOLVE(k) to α. Given an arbitrary clause $\hat{C}$ of size $\ell - 1$, we show that there is a clause $C \in \beta$ with $C \subseteq \hat{C}$. If there is such a C in α we are done since $\alpha \subseteq \beta$. Otherwise let x be a variable such that x and $\bar{x}$ do not belong to $\hat{C}$. Since ALL'(i) holds for α, there are C_1 and C_2 in α, $C_1 \subseteq \hat{C} \vee x$ and $C_2 \subseteq \hat{C} \vee \bar{x}$. If $x \notin C_1$ ($\bar{x} \notin C_2$) then $C = C_1$ ($C = C_2$) will do. Otherwise, after RESOLVE(k), $C \in \beta$ and $C \subseteq \hat{C}$.

$$
\begin{tikzcd}[column sep=small]
 & C \arrow[dl, dash, "x"'] \arrow[dr, dash, "\bar{x}"] & \\
C_1 & & C_2
\end{tikzcd}
$$

This completes the proof of the theorem.

<u>Theorem 5.2</u>: For infinitely many $n \geq 1$, there are unsatisfiable formulas α_n in CN4 over O(n) variables ,(which therefore contain O(n) clauses), such that every resolution tree using α_n contains clauses of size $\geq \frac{1}{2}\sqrt{n}$.

<u>Proof</u>: Let α_n be the set of clauses which correspond to the $k \times k$ square grid. We show that any resolution tree using α_n contains clause of size k. The claim above follows since $n \leq 2k^2$.

Suppose, to the contrary, that there is a resolution tree $\hat{T}$ using α_n with $\max_{i \in \hat{T}} |C_i| = \ell < k$. For every proof tree T using α_n we will assign a <u>positive</u> integer $\mu(T)$ as described below. Starting with $T^{(0)} = \hat{T}$ we show how to transform $T^{(m)}$ into a new proof tree using α_n - $T^{(m+1)}$, such that $\mu(T^{(m+1)}) < \mu(T^{(m)})$. This will establish the contradiction.

Definition of $\mu(T)$

Let i be a vertex in the proof tree T. Define V_i = the set of vertices in $\bar{G}$ which correspond to clauses at the leaves of T_i. Recall that for $V \subseteq \bar{V}$, $\#(V)$ is the number of vertices in $\bar{V}$ which belong to the perimeter of the square.

We say that a vertex i in T satisfies property P (or P(i) is true) iff $k \leq \#(V_i) < 2k$ and for all $j \neq i$ in T_i, $\#(V_j) < k$.

Let ℓ_i be the distance of i from the root of T (1) and let

$$\mu(T) = \sum_{P(i)=\text{true}} \ell_i .$$

By Lemma 4.5 (and the remark following it) there is an i_0 such that $P(i_0)$ is true. Since P(1) is false, $\mu(T) > 0$.

The transformation from $T \equiv T^{(m)}$ to $T' \equiv T^{(m+1)}$

Induction hypothesis: Assume all vertices i with P(i) true and their ancestors contain clauses of size $\leq \ell$ in T (this is obviously true for $T^{(0)}$).

Let i be a vertex in T with P(i) true. By the assumption above, $|C_i| \leq \ell$. By the Regularity Lemma C_i contains a clause C_i' which has a regular proof tree T_i'. We will prove below:

Proposition: For every vertex j in T_i', P(j) is false.

T' is obtained from T by substituting T_i' for T_i in T. Consider all subtrees attached to the path from i to the root of T. They contain all j's such that P(j) = true except for i. During the substitution some of them may be deleted. Those which are not can only get closer to the root. They satisfy P because their subtrees are not changed. By the proposition, for every j in T_i', P(j) is false. There can be at most one vertex j with P(j) false in T and P(j) true in T'. If such a vertex exists it must be a proper ancestor of i. Thus $\mu(T^{(m+1)}) < \mu(T^{(m)})$.

We need to show that the induction hypothesis holds for T'. Consider a vertex j which is an ancestor of a vertex which satisfies P. If j is not on the path from i to the root of T', then C_j' also appeared in T on an ancestor of a vertex which satisfied P (its whol subtree is also a subtree of T). Thus $|C_j'| \leq \ell$ by the induction hypothesis. Otherwise, by the proof of the Substitution Lemma, $C_j' \subseteq C_r$ for some r, an ancestor of i. Thus $|C_j'| \leq |C_r| \leq \ell$ and the induction hypothesis holds for T'. This completes the proof.

Proof of the Proposition

Let V_i (V_i') be the vertices of $\bar{G}$ which correspond to the clause at the leaves of T_i (T_i'). By the Regularity Lemma, $V_i' \subseteq V_i$ and thus $\#(V_i') < 2k$.

By the Correspondence Lemma, C_i' is a regular clause associated with some $G_i' = (V_i', E_i')$. Also, $|F_i^{ext,}| \leq \ell$ since

$$|F_i^{ext,}| \leq |F_i'| = |C_i'| \leq |C_i| \leq \ell.$$

This implies that $\#(V_i') < k$, since otherwise $k \leq \#(V') < 2k$, and so

$$|F_i^{ext,}| \geq k > \ell$$

as in Lemma 4.5 -- a contradiction.

Let j be a vertex in T_i'. $\#(V_j') \leq \#(V_i') < k$ since $V_j' \subseteq V_i'$. Thus j does not satisfy P. This completes the proof of the proposition.

Note that by applying the Regularity Lemma to a given proof tre we obtain a regular proof tree. But the latter can contain longer clauses and thus Theorem 5.2 does not follow directly from the

Regularity Lemma.

Theorem 5.2 answers questions (1) and (2) about the bounded resolution procedures:

Corollary 5.1: BR(k) works for α_n only if $k \geq \frac{1}{2}\sqrt{n}$.

It also answers question (3) about the iterated bounded resolution procedures:

Corollary 5.2: The run-time of IBR1 and IBR2 (IBR3) is greater than $n^{c\sqrt{n}}$ $(2^{c\sqrt{n}})$ for some constant c.

Proof: Corollary 5.1 for BR3 follows from Theorem 5.2. But by Theorem 5.1, the same holds for BR1 and BR2. Corollary 5.2 is obvious for IBR1 and IBR2 since $n^{c\sqrt{n}}$ is a lower bound for the number of distinct clauses of size $\frac{1}{2}\sqrt{n}$. Also, it is easy to see that if IBR3 generates a regular clause associated with some subgraph G, it must generate all regular clauses associated with G. Since a G with $|F_G^{ext}| > \frac{1}{2}\sqrt{n}$ must exist, at least $2^{\frac{1}{2}\sqrt{n}}$ distinct clauses are generated by IBR3.

6. EXPONENTIAL LOWER BOUND FOR REGULAR RESOLUTION

In this section we give a simplified version of Tseitin's proof which establishes an exponential lower bound for regular resolution. As was observed in [3], this implies an exponential lower bound for the DPP. However, an exponential lower bound for DPP may be obtained directly by a much simpler argument. We will sketch this argument at the end of this section.

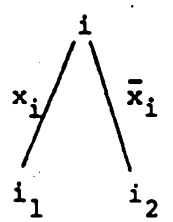
We use again a contradictory system of clauses which corresponds to a graph $\bar{G}$. We consider a regular proof tree T which has the minimal number of distinct clauses. The trick is to define a weight function such that for any clause C which appears on T, $\Sigma w(C) \leq 1$, where the sum is taken over all occurrences of C in T. This will imply

$$\sum_{i \text{ is a vertex in } T} w(C_i) \leq N(T) \equiv \hat{N} .$$

Next, by obtaining a lower bound for the sum above we derive a lower bound for $\hat{N}$.

In order to define the weight function we first check when it is possible that the same clause appears in two different subtrees. The answer appears in Lemma 6.1 which is the hardest part of the proof.

First we need to make several observations. Let i be a vertex on T and let $G_i = (V_i, E_i)$ be the connected component of $\bar{G}$ associated with C_i. If i is not a leaf let i_1 and i_2 be its sons and let x_i be the variable annihilated at i.



Let $G_i' = (V_i', E_i - x_i)$. By Corollary 4.1, $x_i \in E_i$ and there are two possible cases:

<u>case 1</u> x_i breaks a cycle in G_i and $G_{i_1} = G_{i_2} = G_i'$.

<u>case 2</u> x_i splits G_i and G_{i_1} and G_{i_2} are the two connected components of G_i'.

Thus if i is an ancestor of j in T, G_j is a subgraph of G_i. Also, by the observation following the Correspondence Lemma, if $C_{r_1} = C_{r_2}$, then $G_{r_1} = G_{r_2}$.

Hence in case 2 none of the clauses in T_{i_1} can be the same as any clause in T_{i_2}. In what follows we will be interested in the question, "When is it possible that T_{i_1} and T_{i_2} contain the same clause C?"; thus we will be interested in case 1 only. We now consider a subcase of case 1:

Given a subgraph G = (V,E) of G_i we say that <u>x_i breaks an exterior cycle of G_i w.r.t. G</u> if there is a cycle in G_i - G which contains x_i. We will prove later

<u>Lemma 6.1</u>: Assume $C = C_{\ell_1} = C_{\ell_2}$, where ℓ_1 (ℓ_2) is in the left (right) subtree of T_i-- T_{i_1} (T_{i_2}). Then x_i breaks an exterior cycle in G_i w.r.t. G, where $G = G_{\ell_1} = G_{\ell_2}$.

We denote by $\{D_1,\ldots,D_N\}$ the distinct clauses in T and

$S_j = \{i \mid C_i = D_j\}$ for $1 \le j \le \hat{N}$.

Let r be an ancestor of ℓ in T and let $m_{\ell,r}$ be the number of x_p's, with p on the path from r to ℓ, that break an exterior cycle in G_p w.r.t. G_ℓ.

<u>Claim</u>: $\sum_{\ell \in S_j \cap T_r} 2^{-m_{\ell,r}} \le 1$ for $1 \le j \le \hat{N}$.

<u>Proof</u>: An easy induction on the height of T_r. The induction step follows from Lemma 6.1, since whenever D_j appears in both subtrees, $m_{\ell,r}$ grows by one.

Thus $\sum_{\ell \in S_j} 2^{-m_{\ell,1}} \le 1$ for all $1 \le j \le \hat{N}$. (1)

Let n_ℓ be the number of x_p's on the path from 1 to ℓ that break a cycle in G_p (case 1 above).

Let $S_{\bar{G}}(G)$ be the maximum number of edges in the boundary of G that can be deleted from $\bar{G}$ without disconnecting it. We will prove

Lemma 6.2: $n_\ell = m_{\ell,1} + S_{\bar{G}}(G_\ell)$ (2)

Summing up (1) for all j and using (2) yields

$$\hat{N} \geq \sum_\ell 2^{-m_{\ell,1}} = \sum_\ell 2^{-n_\ell} \cdot 2^{S_{\bar{G}}(G_\ell)} \equiv Y_{\bar{G}}(T) \quad (3)$$

A closer look at $Y_{\bar{G}}(T)$ will convince the reader that $Y_{\bar{G}}(T)$ can be defined by the following recursion:

Let $Y_{\bar{G}}(T_i) = 2^{S_{\bar{G}}(G_i)} + \hat{Y}_{\bar{G}}(T_i)$, where

$$\hat{Y}_{\bar{G}}(T_i) = \begin{cases} 0 \text{ if } G_i \text{ consists of a single vertex} \\ Y_{\bar{G}}(T_{i_1}) + Y_{\bar{G}}(T_{i_2}) \text{ if } x_i \text{ splits } G_i \\ \frac{1}{2}(Y_{\bar{G}}(T_{i_1}) + Y_{\bar{G}}(T_{i_2})) \text{ otherwise.} \end{cases}$$

Claim: $Y_{\bar{G}}(T) = Y_{\bar{G}}(T_1)$. (4)

Proof: In the final sum which is obtained from the recursion above, each G_i is represented by $2^{S_{\bar{G}}(G_i)}$. The coefficient of $2^{S_{\bar{G}}(G_\ell)}$ is 2^{-n_ℓ} since we have a factor of 1/2 any time x_i breaks a cycle in G_i.

To get a lower bound for $Y_{\bar{G}}(T_1)$, we define

$$Z_{\bar{G}}(G) = 2^{S_{\bar{G}}(G)} + \min_{a \in E} Z^a_{\bar{G}}(G),$$

where

$$Z^a_{\bar{G}}(G) = \begin{cases} 0 \text{ if } G \text{ consists of a single vertex} \\ Z_{\bar{G}}(G') + Z_{\bar{G}}(G'') \text{ if } a \text{ splits } G \text{ into } G' \text{ and } G'' \\ Z_{\bar{G}}(G') \text{ otherwise, where } G' = (V, E - a) \end{cases}$$

Claim: $Y_{\bar{G}}(T_1) \geq Z_{\bar{G}}(\bar{G})$. (5)

Proof: $Y_{\bar{G}}(T_i) \geq Z_{\bar{G}}(G_i)$ is easily shown by induction on the height o T_i.

Thus, by (3), (4) and (5) we get

$$\hat{N} \geq Z_{\bar{G}}(\bar{G}). \quad (6)$$

But,

$$Z_{\bar{G}}(\bar{G}) = \Sigma 2^{S_{\bar{G}}(G^{(i)})}, \quad (7)$$

where $G^{(i)} = (V^{(i)}, E^{(i)})$ were obtained in the deletion process which is implied by the definition of $Z_{\bar{G}}(\bar{G})$.

If we choose $\bar{G}$ to be the $k \times k$ grid, then by Lemma 4.5 there is an i_0 such that $k < \#(V^{(i_0)}) \leq 2k$. It is easy to check that

$$S_{\bar{G}}(G^{(i_0)}) \geq k - 1. \quad (8)$$

By (6), (7) and (8), $\hat{N} \geq 2^{k-1}$.

Thus we have just proved

Theorem 6.1: For infinitely many $n \geq 1$, there are unsatisfiable formulas α_n in CN4 over $O(n)$ variables (which therefore contain $O(n)$ clauses), such that every regular proof tree using α_n contains at least $2^{\sqrt{n}}$ distinct clauses.

Corollary 6.1: DPP with subsumption is exponential.

Theorems 5.2 and 6.1 can easily be modified to apply to unsatisfiable formulas in CN3 using the $k \times k$ incomplete square grids (see Figure 6.1). However, in the next section, by using more complicated graphs, we will prove stronger results.

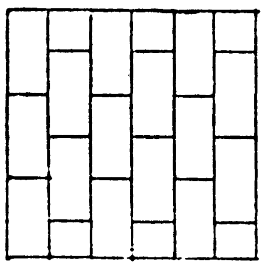


Figure 6.1 The 7×7 incomplete square grid.

Note that by (6) and (7), $\hat{N} \geq \sum_i 2^{S_{\bar{G}}(G^{(i)})}$.

But actually,

$$\hat{N} = \sum_i 2^{S_{\bar{G}}(G^{(i)})}.$$

The second inequality follows from the following observation:

Consider the deletion process implied by the definition of $Z_{\bar{G}}(\bar{G})$. By Lemma 4.2 it has a corresponding regular proof tree $\hat{T}$. We will prove

Lemma 6.3: The number of distinct clauses on $\hat{T}$ associated with $G^{(i)}$ is $\leq 2^{S_{\bar{G}}(G^{(i)})}$. Thus $N(\hat{T}) \leq \sum_i 2^{S_{\bar{G}}(G^{(i)})}$, and $\hat{T}$ must be optimal.

Note that $\hat{T}$ is a DP tree. Thus, for systems of clauses associated with graphs, regular resolution is no more powerful than the Davis Putnam procedure. Nevertheless, the direct proof for Corollary 6.1 is much simpler than the proof of Theorem 6.1. We give an outline of this proof (exponential lower bound for DPP) at the end of this section.

Proof of Lemma 6.1: As we observed above, x_i must break a cycle in G_i. x_i does not belong to the boundary of G, since if it did, then because the tree is regular, the Correspondence Lemma would imply that $x_i \in C_{\ell_1}$ and $\bar{x}_i \in C_{\ell_2}$ -- a contradiction ($C_{\ell_1} = C_{\ell_2}$).

Assume x_i does not break an exterior cycle in G_i w.r.t. G. Then one of the connected components of $G_i - G$ must be split in two by deleting x_i:

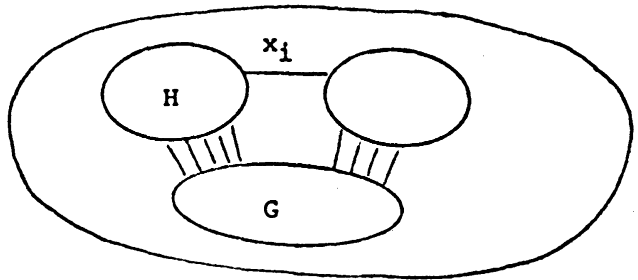


Figure 6.2 G_i

Let H be one of the two.

For r on the path from i_1 to ℓ_1, let $H_r = H \cap G_r$ and let $\delta_r = \sum_{\substack{z_m \in F^{ext}_{H_r} \\ z_m^{\varepsilon_m} \in C_r}} (1 - \varepsilon_m)$, i.e. δ_r is the contribution to the parity of C_r by edges in the exterior boundary of H_r.

<u>Claim</u>: $\delta_{i_1} = \delta_{\ell_1} + \varepsilon(H)$. (9)

Similarly, $\delta_{i_2} = \delta_{\ell_2} + \varepsilon(H)$. But $\delta_{\ell_1} = \delta_{\ell_2}$ and $\delta_{i_1} \neq \delta_{i_2}$.

($C_{i_1} = C_i \vee x_i$, $C_{i_2} = C_i \vee \bar{x}_i$ and $x_i \in F^{ext}_{H_{i_1}} = F^{ext}_{H_{i_2}}$).

This contradiction proves the lemma.

To prove the claim we show by induction on the distance of r from i_1 that

$$\delta_{i_1} = \delta_r + \varepsilon(H - H_r). \qquad (10)$$

This is obviously true for $r = i_1$. Assume that the result holds for r, and let r_1, r_2 be the sons of r, r_1 is on the path to ℓ_1. The only case in which δ_r (or $\varepsilon(H - H_r)$) can be different from δ_{r_1} (or $\varepsilon(H - H_{r_1})$, respectively) is when the deletion of x_r from G_r splits it into G_{r_1} and $G_{r_2} \subseteq H_r$.

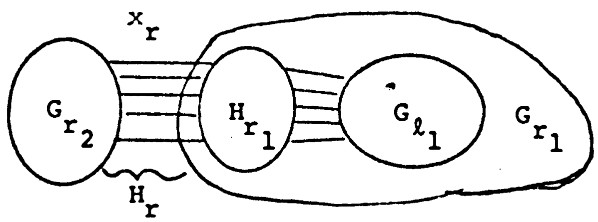


Figure 6.3 G_r

δ_r has some terms deleted and some new terms added. Since in mod 2 sum "subtracting" is the same as "adding",

$$\delta_{r_1} = \delta_r + \underbrace{\text{terms added + terms deleted}}_{\lambda_r}. \qquad (11)$$

But

$$\lambda_r = \varepsilon(G_{r_2}). \tag{12}$$

This follows from

a) $\lambda_r = \sum_{x_i \in F^{ext}_{G_{r_2}}} (1 - \varepsilon_i')$,

b) $\varepsilon(G_{r_2}) \neq \delta(C_{r_2}) = \sum_{x_i \in F^{ext}_{G_{r_2}}} (1 - \varepsilon_i)$,

and

c) $\varepsilon_r' \neq \varepsilon_r$ and $\varepsilon_i' = \varepsilon_i$ for all $i \neq r$.

Also, $H - H_r = H - H_{r_1} \cup G_{r_2}$ and the two sets are disjoint. Thus

$$\varepsilon(H - H_r) = \varepsilon(H - H_{r_1}) + \varepsilon(G_{r_2}). \tag{13}$$

Substituting (11) — (13) into (10) yields (10) for r_1. ∎

<u>Proof of Lemma 6.2:</u>

Let $\{H_j^{(i)}\}_{j=1}^{m_i}$ be the connected components of $G_i - G_\ell$, and let $k_j^{(i)}$ be the number of edges which connect vertices in G_ℓ to vertices in $H_j^{(i)}$. By definition of $S_{G_i}(G_\ell)$,

$$S_{G_i}(G_\ell) = |F^{int}_{\ell,i}| + \sum_{j=1}^{m_i} (k_j^{(i)} - 1) \tag{14}$$

where $F^{int}_{\ell,i}$ is the interior boundary of G_ℓ w.r.t. G_i.

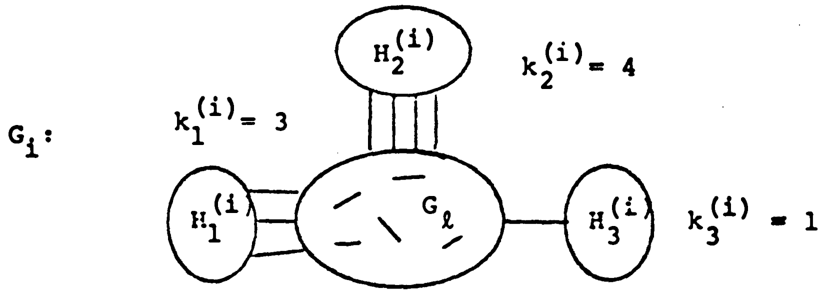


Figure 6.4 $S_{G_i}(G_\ell) = 10$

Claim: $S_{G_i}(G_\ell)$ is the number of variables x_j from i to ℓ that break a cycle in G_j, which is not an exterior cycle with respect to G_ℓ.

The claim proves the lemma since for $i = 1$, $G_1 = \bar{G}$, and hence the number of times x_j breaks a cycle in G_j which is not an exterior cycle w.r.t. G_ℓ is $n_\ell - m_{\ell,1}$.

The claim is easily proved by induction on the distance of i from ℓ. For $i = \ell$, $0 = 0$. Assume it holds for i_1 the son of i. If x_i splits G_i, all the terms for i_1 in (14) are the same as for i (also if x_i is in the boundary of G, since the corresponding k_j is 1 and does not contribute anything to the sum in (14)). If x_i breaks an exterior cycle in G_i w.r.t. G_ℓ nothing changes either. Otherwise,

a) if $x_i \in F^{int}$, then $|F^{int}_{\ell,i}| = |F^{int}_{\ell,i_1}| + 1$

b) if $x_i \in F^{ext}$, then

$$\sum_{j=1}^{m_i=m_{i_1}} (k_j^{(i)}-1) = \sum_{j=1}^{m_{i_1}} (k_j^{i_1} - 1) + 1$$

and

c) if x_i is as in Figure 6.2 (the remaining case), then

$$\sum_{j=1}^{m_i=m_{i_1}-1} (k_j^{(i)}- 1) = \sum_{j=1}^{m_{i_1}} (k_j^{(i_1)}- 1) + 1.$$

In b) one of the $k_j^{(i_1)}$'s increases by 1 and in c) one of the $k_j^{(i)}$'s splits into two : $k_{j_1}^{(i_1)}$ and $k_{j_2}^{(i_1)}$, and the one is subtracted twice. Thus the claim holds for i.

Proof of Lemma 6.3:

It is implicit in the proof of Lemma 6.1 that if i is an ancestor of ℓ in a proof tree, then the parity of the number of complemented variables corresponding to edges which connect G_ℓ with any one of the connected components of $G_i - G_\ell$ is determined given G_i and G_ℓ and does not depend on the path from i to ℓ (this is exactly the meaning of (9)). Thus taking $i = 1$, $G^{(\ell)}$ can have at most

$$2^{|F_\ell^{int}| + \sum_j (k_j - 1)} = 2^{S_{\bar{G}}(G_\ell)}$$

distinct clauses associated with it (2^{k_j-1} is exactly half of the ways to assign complements to the edges which connect $G^{(\ell)}$ and H_j. Indeed, exactly half of them satisfy the parity mentioned above). ∎

We now outline a direct proof of an exponential lower bound for the Davis Putnam procedure. It goes as follows:

1. Proof of the Correspondence Lemma.
2. Showing the existence of G_i, such that $S_{\bar{G}}(G_i) \geq k-1$.
3. Showing that at least $2^{S_{\bar{G}}(G_i)}$ distinct regular clauses associated with G_i must be present.

1 and 2 are the same as for regular resolution. For 3 we need only a few observations:

a) All clauses generated by DPP are regular;

b) Without subsumption, if a clause C is generated and it is associated with the subgraph G_r then all regular clauses associated with G are generated at the same stage;

c) To count how many clauses which correspond to G_i are generated we may assume w.l.o.g. that we use DPP without subsumption and count all the clauses C which correspond to G_i and are not subsumed by any other clause at the time they are all generated.

d) If C corresponds to G_i and is subsumed by C', then C' is a regular clause associated with a connected component of $\bar{G} - G_i$

(since $x \in F_i \Longrightarrow x$ has not been eleminated yet $\Longrightarrow x \notin C'$).

e) If $\{H_j\}_{j=1}^{m}$ are the connected components of $\bar{G} - G_i$ and k_j is the number of edges connecting G_i and H_j, then $S_{\bar{G}}(G_i) = |F_i^{int}| + \sum_{j=1}^{m} (k_j - 1)$.

f) If C is a clause associated with G_i such that for each j the unique clause $C_j \subseteq C$ which is associated with H_j satisfies

$\delta(C_j) = \varepsilon(H_j)$, then C is a regular clause associated with G_i and is not subsumed by any clause when it is generated.

(By d), C can be subsumed only by some C_j. But by a) none of the C_j is generated. C is regular by a parity argument.)

g) The number of such C's is

$$2^{|F_i^{int}| + \sum_{j=1}^{m}(k_j-1)} = 2^{S_{\bar{G}}(G_i)}. \qquad \blacksquare$$

The proof of an exponential lower bound for the DPP is much simpler than the proof of the same result for regular results, since the latter is much stronger. This is because 1) DPP is a special case of regular resolution, and 2) in the case of regular resolution we count only distinct clauses which appear on the proof tree. It is not at all clear from the direct proof outlined above how many of the $2^{\sqrt{n}}$ distinct clauses generated by the DPP must appear in a DP tree.

7. SOME STRONGER RESULTS

The choice of the k × k square grid for $\bar{G}$ has yielded all the results in the preceding sections. This choice was not crucial and we will see later that a better, but slightly more complicated, choice can be made.

Let $\bar{G}$ be a given graph and let ρ be a deletion process defined on $\bar{G}$.

Let $f(\bar{G},\rho) = \max_i |F_i|$

$s(\bar{G},\rho) = \max_i s_{\bar{G}}(G_i)$

where the maximum is taken over all different graphs G_i obtained during the deletion process ρ.

Let $f(\bar{G}) = \min_\rho f(\bar{G},\rho)$ and $s(\bar{G}) = \min_\rho s(\bar{G},\rho)$.

Let $f_n = \max_{\bar{G}\in\mathscr{D}_n} f(\bar{G})$

$s_n = \max_{\bar{G}\in\mathscr{D}_n} s(\bar{G})$

where $\mathscr{D}_n$ is the class of all connected graphs with n edges and degree 3. By applying minor modifications to the proof of Theorem 6.1 and by the one to one correspondence between deletion processes and regular proof trees (Lemmas 4.1 and 4.2) one obtains that

1) f_n is a lower bound for the size of clauses in a regular proof tree.

2) 2^{s_n} is a lower bound for the number of distinct clauses in a regular proof tree.

Obviously $s_n \leq f_n$. We will prove

<u>Lemma 7.1</u>: $s_n \geq (f_n - 1)/2$.

Thus, the two parameters are linearly related. We denote by $\hat{f}_n$ the lower bound on the size of clauses in any proof tree for the set of clauses associated with some $\bar{G} \in \mathscr{D}_n$. Obviously $\hat{f}_n \leq f_n$.

The choice of the $k \times k$ square grid was made by Tseitin. It yields s_n and $\hat{f}_n$, both of order $\sqrt{n}$. A more complicated family of graphs [7] yields s_n and $\hat{f}_n$ both of order $n/(\log n)^2$:

Let $\bar{G} = (\bar{V}, \bar{E})$ be the k-dimensional cube, i.e.

$\bar{V} = \{\lambda = (\lambda_1, \ldots, \lambda_k) \mid \lambda_i \in (0,1)\}$

$\bar{E} = \{(\lambda,\mu) \mid$ For exactly one j, $\lambda_j \neq \mu_j\}$

$\bar{G}$ has 2^k vertices and $n = k2^{k-1}$ edges.

Note that the degree of $\bar{G}$ is $k = O(\log n)$. We now transform $\bar{G}$ into a new graph G with degree 3 ($G \in \mathcal{Y}_n$):

$G = (V, E)$

$V = \{\lambda^{(i)} \mid 1 \leq i \leq k, \lambda \in \bar{V}\}$

$E = \{(\lambda^{(i)}, \mu^{(i)}) \mid (\lambda,\mu) \in \bar{E} \quad \lambda_i \neq \mu_i\} \cup \{(\lambda^{(i)}, \lambda^{(i+1 \bmod k)}) \mid$
$\lambda \in \bar{V}\}$.

The transformation is indicated by Figure 7.1.

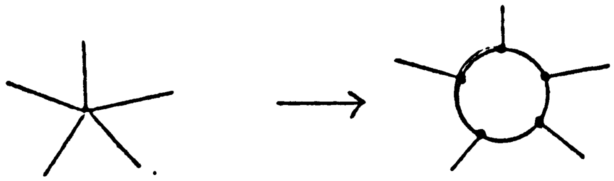


Figure 7.1 (k = 5)

G has $n = 1.5k2^k$ edges and $k2^k$ vertices. Let G_ℓ be a connected subgraph of G. $\bar{G}_\ell = (\bar{V}_\ell, \bar{E}_\ell)$ is the <u>projection</u> of G_ℓ onto $\bar{G}$ if

$\bar{V}_\ell = \{\lambda \mid \lambda^{(i)} \in V_\ell$ for some i$\}$

$\bar{E}_\ell = \{(\lambda,\mu) \mid (\lambda^{(i)}, \mu^{(i)}) \in E$ for some i$\}$.

Let ρ be a deletion process on G. As in Lemma 4.5, there must be some i_0 such that $2^{k-2} \leq |\bar{V}_{i_0}| \leq 2^{k-1}$, where $G_{i_0} = (V_{i_0}, E_{i_0})$ is one of the subgraphs of G which is generated in the deletion process, and $\bar{G}_{i_0} = (\bar{V}_{i_0}, \bar{E}_{i_0})$ is its projection on $\bar{G}$. We will prove

<u>Lemma 7.2</u>: $|\bar{F}_{i_0}^{ext}| \geq 2^{k-2}$.

But $|F_{i_0}^{ext}| \geq |\bar{F}_{i_0}^{ext}|/k$, since if any of the k edges incident with λ is in $\bar{F}_{i_0}^{ext}$, then at least one of the edges incident with some $\lambda^{(i)}$ is in $F_{i_0}^{ext}$. Together with Lemma 7.2 we have, for all ρ,

$$f(G\ ,\rho) \geq 2^{k-2}/k = n/6k^2 .$$

Thus $f_n \geq f(G.) \geq O(n/(\log n)^2)$. To show that actually $\hat{f}_n \geq O(n/(\log n)$ we need to repeat the proof of Theorem 6.2. It is necessary to use a modified property P: a vertex i on a proof tree T satisfies property P' if $2^{k-2} \leq |\bar{V}_i| < 2^{k-1}$ and for every proper descendant of i, j, $|\bar{V}_j| < 2^{k-2}$ (where for every ℓ $\bar{V}_\ell$ is the projection of V_ℓ and V_ℓ is the set of vertices which corresponds to the clauses on the leaves of T_ℓ). Lemmas 7.2 and 7.3 are used instead of Lemma 4.5. We leave the details to the interested reader.

The discussion above proves

<u>Theorem 7.1</u>: For infinitely many $n \geq 1$, there are unsatisfiable formulas α_n in CN3 over O(n) variables (which therefore contain O(n) clauses) such that any resolution proof tree using α_n contains clauses of size $\geq n/(\log n)^2$.

Corollaries 5.1 and 5.2 are transformed similarly.

<u>Theorem 7.2</u>: For infinitely many $n \geq 1$ there are unsatisfiable formulas α_n in CN3 over O(n) variables (which therefore contain O(n) clauses) such that any regular proof tree using α_n contains $2^{n/(\log n)^2}$ distinct clauses.

Note that the length of the input is $N = O(n\log n)$ and the lower bound in Theorem 7.1 is $N/(\log N)^3$ and in Theorem 7.2, $2^{N/(\log N)^3}$. The best bounds using systems of clauses corresponding to graphs would be n ($N/\log N$) and 2^n ($2^{N/\log N}$) respectively in case there are infinitely many $n \geq 1$ such that $\hat{f}_n = O(n)$. Thus Theorems 7.1 and 7.2 give nearly the best bounds one can expect using these techniques.

Proof of Lemma 7.1

We show that

$$s(\bar{G}) \geq (f(\bar{G}) - 1)/2 \tag{1}$$

for all $\bar{G}$ with n edges and degree 3. This implies that $s_n \geq (f_n - 1)/2$.

case 1 $\bar{G}$ is biconnected.

We show that for every subgraph G of $\bar{G}$,

$$s_{\bar{G}}(G) \geq |F|/2, \tag{2}$$

which implies (1) above. Let $\{H_i\}_{i=1}^r$ be the connected components of $\bar{G} - G$. Let k_i be the number of edges in F connecting a vertex in H_i to a vertex in G.

$$s_{\bar{G}}(G) = |F^{int}| + \sum_{i=1}^{r} (k_i - 1).$$

But $|F^{ext}| = \sum_{i=1}^{r} k_i$ and $k_i \geq 2$ since $\bar{G}$ is biconnected.

Hence $s_{\bar{G}}(G) \geq \frac{1}{2}|F|$, i.e. (2) holds. This completes the proof of case 1.

case 2 Let $\{H_j\}_{j=1}^r$ be the biconnected components of $\bar{G}$.

Obviously,

$$s(\bar{G}) = \max_j s(H_j). \tag{3}$$

Also,

$$f(\bar{G}) \leq \max_j f(H_j) + 1. \tag{4}$$

The latter is obtained by choosing a special deletion process ρ:

Let ρ_j be an optimal deletion process for H_j (i.e. for which $f(H_j)$ is achieved). Let ρ be the process of applying ρ_1. If it happens that G_i splits into G_{i_1} and G_{i_2} and G_{i_1} contains only edges outside of H_1, then G_{i_1} consists of $H_{j_1} \cup \ldots \cup H_{j_k}$ plus a <u>bridge</u> -- an edge from one of them, say H_{j_1}, to H_1.

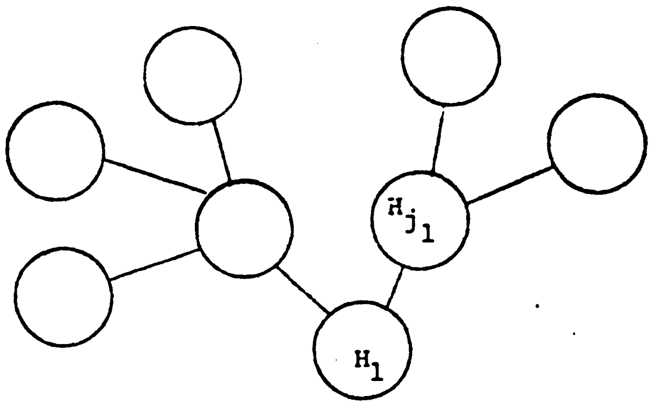


Figure 7.2 Each circle is some H_j and each line is some bridge.

We must specify the process for G_{i_1}: ρ deletes the bridge and applies ρ_{j_1}. This is repeated each time the union of some biconnected components plus a bridge is separated from the component whose edges are currently deleted. It follows that

$$f(\bar{G},\rho) = \max_j f(H_j) + 1$$

from which (4) follows.

(3), (4) and <u>case 1</u> ((2)) imply (1):

$$s(\bar{G}) = \max_j s(H_j) \geq \frac{1}{2}\max_j f(H_j) \geq \frac{1}{2}(f(\bar{G}) - 1). \quad \blacksquare$$

<u>Proof of Lemma 7.2</u>

We show by induction on k that for any subgraph $G = (V,E)$ of $\bar{G}_n$, if $|V| = 2^{k-1} + r$ then $|F^{ext}| \geq 2^{k-1} - |r|$. This will prove the lemma since $|V_{i_0}| \le 2^{k-1}$, $r \le 0$ and thus $|F_{i_0}^{ext}| \ge |V_{i_0}| \ge 2^{k-2}$.

Basis is trivial.

Induction step: Let $|V| = 2^{k-1} + r$. Let $G_{(0)}$ be the projection of G onto $\lambda_k = 0$; i.e. $G_{(0)} = (V_{(0)}, E_{(0)})$ and

$$V_{(0)} = \{\lambda \mid \lambda \in V,\ \lambda_k = 0\}$$

$$E_{(0)} = \{(\lambda,\mu) \mid (\lambda,\mu) \in E,\ \lambda,\mu \in V_{(0)}\}.$$

Similarly, define $G_{(1)}$ to be the projection of G on $\lambda_k = 1$. $|V_{(0)}| = 2^{k-2} + r_0$ and $|V_{(1)}| = 2^{k-2} + r_1$. By the induction hypothesis,

$$|F^{ext}_{(0)}| \ge 2^{k-2} - |r_0| \quad \text{and} \quad |F^{ext}_{(1)}| \ge 2^{k-2} - |r_1| .$$

If r_0 and r_1 are of the same sign, then by adding the two inequalities, we get

$$|F^{ext}| \ge |F^{ext}_{(0)}| + |F^{ext}_{(1)}| \ge 2^{k-1} - |r|.$$

The other cases are, w.l.o.g.

(1) $r > 0$, $r_0 > 0$, $r_1 = r - r_0 < 0$ and

(2) $r < 0$, $r_0 > 0$, $r_1 = r - r_0 < 0$.

In these cases there are at least $r_0 - (r - r_0) = 2r_0 - r$ edges in F^{ext} going from $V_{(0)}$ to $V_{(1)}$. Thus

$$|F^{ext}| \ge |F^{ext}_{(0)}| + |F^{ext}_{(1)}| + 2r_0 - r \ \ge 2^{k-1} + 2r_0 - r - |r_0| \ - |r_1|$$

$$= 2^{k-1} + 2r_0 - r - r_0 + r - r_0 = 2^{k-1} \ge 2^{k-1} - |r|.$$

This completes the proof of the lemma.

8. THE POWER OF EXTENDED RESOLUTION

For three literals x, y, z and a boolean function f over 2 variables, let $\alpha(f,z,x,y)$ be the set of clauses over x, y, and z which is equivalent to $z \equiv f(x,y)$. One can verify that $\alpha(f,z,x,y)$

exists for all 16 possible choices of f.

Example:

$\alpha(V,z,x,y) = \{(\bar{x}Vz),(\bar{y}Vz),(\bar{z}VxVy)\}$

$\alpha(+,z,x,y) = \{(\bar{x}VyVz),(xV\bar{y}Vz),(xVyV\bar{z}),(\bar{x}V\bar{y}V\bar{z})\}$

$\alpha(U_1,z,x,y) = \{(\bar{z}Vx),(\bar{x}Vz)\}$ where $U_1(x,y) = x$.

The extension rule is applied by choosing f,x and y, creating a new variable z, and adding $\alpha(f,z,x,y)$. Extended Resolution is resolution augmented with the extension rule.

In the preceding sections we considered several restricted versions of resolution. We saw that

1) Allowing only clauses of size up to a certain constant bound causes the procedure to not always work correctly.

2) Allowing only regular resolution enabled us to prove an exponential lower bound. The question which naturally arises is: what happens if we impose these restrictions on extended resolution? The following theorems answer this question.

Let T be a resolution proof tree (with or without extension). Let N = N(T) and m be the number of variables in the clauses on T.

Theorem 8.1: We can transform T into a regular proof tree T' which contains at most $O(N^2m)$ distinct clauses.

If the original set of clauses has at most three literals per clause, then

Theorem 8.2: We can transform T into a proof tree T' with clauses of size ≤ 3. T contains at most O(Nm) distinct clauses.

Proof of Theorem 8.1

In order to transform T into a regular tree we can eliminate all irregularities using renaming.

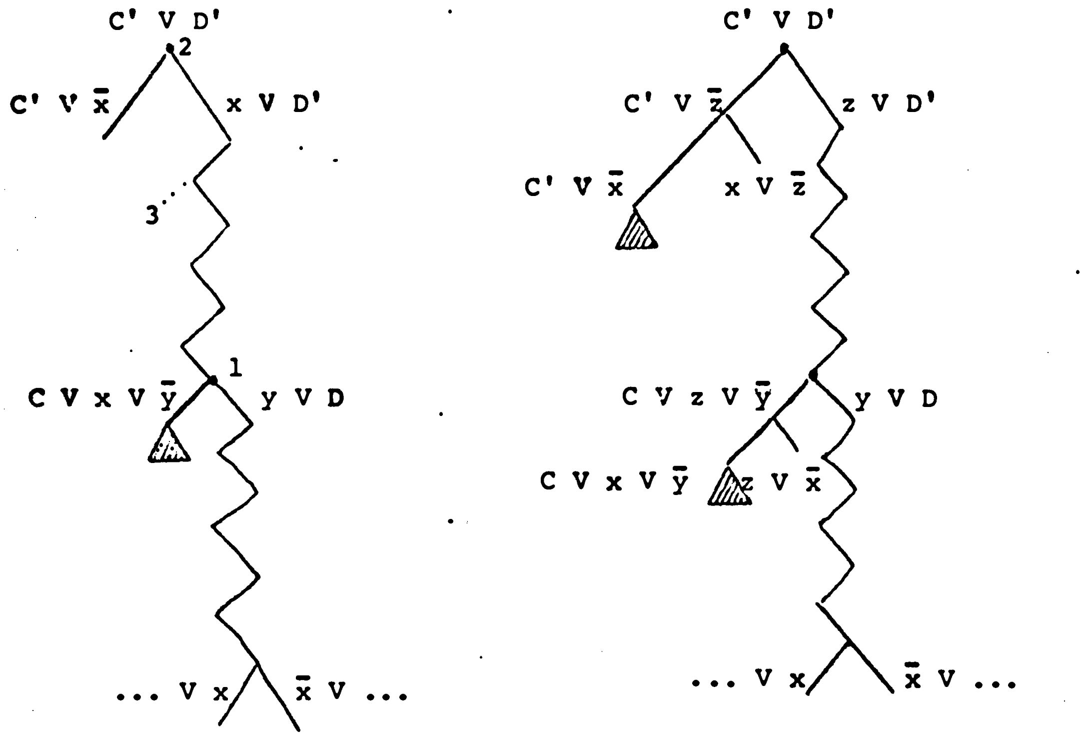


Figure 8.1

i.e. we prevent reintroducing x by substituting z for it. We do the same whenever x is reintroduced between 1 and 2 (e.g. at 3). Later, at the point where x is annihilated again (at 2) we substitute $\bar{z}$ for $\bar{x}$ and annihilate z.

At first glance it is not at all clear that

1) By using renaming we can eliminate all irregularities.

2) That in case renaming succeeds it does not introduce too many new clauses (The number of new clauses might be O(N); i.e. one renaming might double the number of distinct clauses.)

We will use renaming, but in a careful manner.

w.l.o.g. assume that each distinct clause in T is derived by the same subtree in each of its occurrences in T. This implies that the same clause cannot appear twice on any path from the root to one of the leaves.

Define the following partial order on the clauses on T: C < D iff D appears at a root of a subtree which contains C. Our assumption on T implies that < is indeed a partial order. Thus we can number the clauses $C_1, C_2, \ldots, C_N$, so that $C_i < C_j \Rightarrow i < j$ (topological sort).

We use Nm variables: for each original variable x we use $x^{(1)}, \ldots, x^{(N)}$ where $x^{(1)}$ is identical with x. $x^{(i)}$ is said to be the i-th level of x. A clause is at level i if all its literals are at level i. The new clauses formalize

$$x^{(i+1)} \equiv x^{(i)} \text{ for } 1 \leq i \leq N-1.$$

If $C = x^{(i)} \vee D$ we <u>rename $x^{(i)}$ in C</u> by

$$\begin{array}{ccc} & x^{(i+1)} \vee D & \\ / & & \backslash \\ x^{(i)} \vee D & & \bar{x}^{(i)} \vee x^{(i+1)} \end{array}$$

Similarly, we can rename $\bar{x}^{(i)}$ in C. We <u>rename C at level i to level i</u>

if $\quad C = {x_{i_1}^{(i)}}^{\varepsilon_1} \vee \ldots \vee {x_{i_k}^{(i)}}^{\varepsilon_k}$, $1 \leq i_1 < i_2 < \ldots < i_k \leq m$

and we rename ${x_{i_1}^{(i)}}^{\varepsilon_1}$, then ${x_{i_2}^{(i)}}^{\varepsilon_2}$ and so forth. We obtain

$$C' = {x_{i_1}^{(i+1)}}^{\varepsilon_1} \ldots {x_{i_k}^{(i+1)}}^{\varepsilon_k}$$

at level i+1. Note that C is renamed in a fixed order.

We <u>rename α from level i to level $j > i$</u> by renaming it to level $i+1, i+2, \ldots, j$.

All the original clauses are at level 1. We use the following inductive construction: if C_i is on a leaf of T, i.e. it is an original clause, then we rename C_i from level 1 to level i. If

$$\begin{tikzcd}[column sep=small] & C_i \arrow[dl, dash] \arrow[dr, dash] & \\ C_j & & C_k \end{tikzcd}$$

is in T, then $j < i$ and $k < i$ and we rename C_j and C_k from levels j and k respectively to level i. Then we resolve them to get C_i at level i.

The new tree T' is regular and contains $\leq O(N^2 m)$ distinct clauses. ∎

<u>Proof of Theorem 8.2</u>

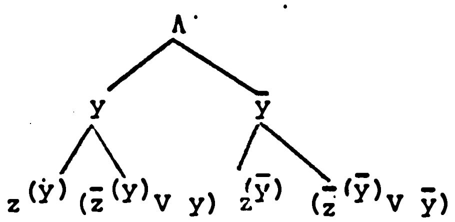
For every clause C in T we have a new variable $z^{(C)}$ such that $z^{(C)} \equiv C$. Using additional variables, we show that if

$$\begin{tikzcd} & E \arrow[dl, dash, "x"'] \arrow[dr, dash, "\bar{x}"] & \\ C & & D \end{tikzcd}$$

is in the tree, then given trees for $z^{(C)}$ and $z^{(D)}$, we can derive $z^{(E)}$. If the top of T looks like

$$\begin{tikzcd}[column sep=small] & \Lambda \arrow[dl, dash] \arrow[dr, dash] & \\ y & & \bar{y} \end{tikzcd}$$

then it follows by induction from the claim above that we can derive $z^{(y)}$ and $z^{(\bar{y})}$. But $z^{(y)} \equiv y$ and $z^{\bar{y}} \equiv \bar{y}$, and using the available clauses expressing these facts we get the top of T':



Since we will use only clauses of size ≤ 3, this will complete the construction of T'.

Let $x_1, x_2, \ldots, x_m$ be the variables used in T. Assume

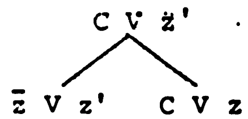
$C = y_{i_1} \vee \ldots \vee y_{i_k}$

where $1 \leq i_1 \leq \ldots \leq i_k \leq m$. Let $C_j \equiv y_{i_1} \vee \ldots \vee y_{i_j}$ $(C_k = C)$. Thus

$$z^{(C_1)} \equiv y_{i_1} \tag{1}$$

and $z^{(C_j)} \equiv z^{(C_{j-1})} \vee y_{i_j}$. (For the corresponding set of clauses use the examples in the beginning of the section.)

We now show how to obtain $z^{(E)}$ from $z^{(C)}$ and $z^{(D)}$. Let $C = u_1 \vee \ldots \vee u_r$, $D = v_1 \vee \ldots \vee v_s$, x is u_p, $\bar{x}$ is v_q. We denote by $C \vee z \to C \vee z'$ the subtree



and let $\overset{*}{\to}$ be the transitive closure of $\to$. Note that if $w \in E$, then $w \vee C' \overset{*}{\to} z^{(E)} \vee C'$: If w is the j-th literal in E then

$$w \vee C' \to z^{(E_j)} \vee C' \to z^{(E_{j+1})} \vee C' \to \ldots \to z^{(E)} \vee C'$$

(the clauses $\bar{w} \vee z^{(E_j)}$, $\bar{z}^{(E_j)} \vee z^{(E_{j+1})}$ are available from (1)).

We decompose $z^{(C)}$ (i.e. $z^{(C_r)}$) as follows:

$$
\begin{array}{ccc}
z^{(C_r)} & & \bar{z}^{(C_r)} \vee u_r \vee z^{(C_{r-1})} \\
& \searrow \swarrow & \\
& z^{(C_{r-1})} \vee u_r \overset{*}{\to} z^{(C_{r-1})} \vee z^{(E)} & \qquad \bar{z}^{(C_{r-1})} \vee u_{r-1} \vee z^{(C_{r-2})}
\end{array}
$$

$$
\begin{array}{ccc}
z^{(C_{r-1})} \vee z^{(E)} & & \bar{z}^{(C_{r-1})} \vee u_{r-1} \vee z^{(C_{r-2})} \\
& \searrow \swarrow & \\
& z^{(C_{r-2})} \vee z^{(E)} \vee u_{r-1} \overset{*}{\to} z^{(C_{r-2})} \vee z^{(E)} . &
\end{array}
$$

$$\cdots$$

since $u_r, u_{r-1} \ldots \in E$.

Thus by repeating the above we derive $z^{(C_{p-1})} \vee z^{(E)} \vee x$ and $z^{(D_{q-1})} \vee z^{(E)} \vee \bar{x}$. We resolve them to get $z^{(C_{p-1})} \vee z^{(D_{q-1})} \vee z^{(E)}$. If we continue as before we will get clauses of size 4. We define a new variable $\hat{z} \equiv z^{(D_{q-1})} \vee z^{(E)}$. We use the new clauses to derive $z^{(C_{p-1})} \vee z^{(D_{q-1})} \vee z^{(E)} \overset{*}{\to} z^{(C_{p-1})} \vee \hat{z}$. Then as before we continue the decomposition to get $z^{(C_{p-2})} \vee z^{(E)} \vee \hat{z}$; then "substituting" back $\hat{z}$ we get

$$
\begin{array}{ccc}
z^{(C_{p-2})} \vee z^{(E)} \vee \hat{z} & & \bar{\hat{z}} \vee z^{(E)} \vee z^{(D_{q-1})} \\
& \searrow \swarrow & \\
& z^{(C_{p-2})} \vee z^{(E)} \vee z^{(D_{q-1})} . &
\end{array}
$$

This process (using new $\hat{z}$'s) is repeated. The reader can easily see that $z^{(E)}$ is obtained at the end. ∎

A similar proof shows that any tree T for a set of clauses with at most $\ell \geq 3$ literals per clause can be transformed to a tree T' such that all the clauses in T' are of size $\leq \ell$.

Another property which shows that extended resolution is very powerful was first observed by Tseitin:

<u>Theorem 8.3</u>: For any set of clauses which corresponds to a graph there is a resolution proof tree T, which uses extension, such that the number of distinct clauses on T is polynomial.

One way to prove it is to consider a deletion process and to choose one subgraph for every set of subgraphs which have the same exterior boundary. We denote by $G_{\bar{i}}$ the subgraph which represents G_i and introduce a new variable $z_{\bar{i}}$, where $z_{\bar{i}} \equiv$ sum modulo 2 of the edge labels of $F_{\bar{i}}^{ext}$. One can show that using the tricks of Theorem 8.2 with $+$ instead of $\vee$, there is a proof tree for $z_{\bar{i}}^{\varepsilon_{\bar{i}}}$ where $\varepsilon_{\bar{i}}$ is the parity of G_i; i.e. if x_r splits G_r into G_{r_1} and G_{r_2} then one can construct a proof tree for $z_{\bar{r}}^{\varepsilon_{\bar{r}}}$ using $z_{\bar{r}_1}^{\varepsilon_{\bar{r}_1}}$ and $z_{\bar{r}_2}^{\varepsilon_{\bar{r}_2}}$. The contradiction is derived from the decomposition of z_p and $\bar{z}_q$ where $G_p \cup G_q$ is the whole graph and G_p (G_q) has parity 1 (0). If the graph has O(n) vertices and edges this technique yields a tree with $O(n^2)$ distinct clauses. A proof tree with O(nlogn) clauses is obtained if we use graphs which are not necessarily connected. This is achieved by taking a full binary tree, G_1 = the given graph and if i_1 and i_2 are the sons of i, then G_{i_1} (G_{i_2}) consists of exactly half the vertices of G_i and all the edges in $\bar{G}$ among them. The complete proof requires the (obvious) definition of boundary and parity of unconnected graphs.

Tseitin has also observed that using extension one can transform any boolean formula α into a formula β in CNF such that

$$|\beta| \leq O(|\alpha|).$$

On the other hand, we know that without introducing new variables there are formulas α such that the equivalent β in CNF must be of length exponential in $|\alpha|$.

9. CONCLUSION AND OPEN PROBLEMS

We have considered several procedures for determining whether a set of clauses is contradictory (a formula in DNF is a tautology), based on the resolution rule: 1) The Davis Putnam procedure; 2) regular resolution; 3) unrestricted resolution; 4) extended resolution; and 5) three versions of (iterated) bounded resolution. The first 4 are nondeterministic and have increasing computational power; i.e., each procedure is a special case of the next. They are nondeterministic since in the case of DPP the choice of the eliminated variable is not specified and in the other cases the way to construct the proof tree is not specified. The bounded resolution versions are all deterministic and do not fit into this "hierarchy". We have shown that the DPP, regular resolution and iterated bounded resolution have exponential complexity, while bounded resolution does not always work. It still remains to determine:

1) what is the complexity of extended resolution?

and 2) what is the complexity of unrestricted resolution?

As a first step one might try to prove a nonlinear lower bound for these procedures. The fact that the first four procedures are nondeterministic should not matter: as was pointed out in the introduction, if we found a nondeterministic procedure of polynomial complexity for the tautology problem, then this would settle an important open problem (namely: is NP closed under complementation?). The fact that some of the deterministic procedures are of exponential complexity and that bounded resolution does not always work does not imply that they do not deserve more investigation. If $P \neq NP$, which many researchers believe is the case, then no procedure for the tautology problem can run in polynomial time. In this case, as was pointed out by M. Rabin [9], we must be satisfied with procedures which sometimes give incorrect answers. In this light the bounded resolution procedures seem to be fairly good heuristic procedures for the tautology problem. Also, some procedures which are exponential might be polynomial for some large nontrivial class of inputs. In [6], such a class was found for the DPP. In our paper we exploited Tseitin's ingenious examples, i.e. the set of clauses corresponding to graphs. We found out that for these sets of clauses, the best regular tree is a DP tree. One might ask:

3) are there examples for which DPP generates many more distinct clauses than regular resolution?

The same question is also open for

4) regular resolution versus unrestricted resolution

and for

5) unrestricted resolution versus extended resolution.

We only know that the answer is positive for regular resolution versus extended resolution. However we conjecture that this is the case for questions 3) and 5). To answer question 3 one will need a new contradictory set of clauses, other than those corresponding to graphs. It is interesting to note that 5) is open although extended resolution was shown in Section 8 to be much more powerful than unrestricted resolution in many aspects. We feel that the techniques we used, especially the lemmas in Section 4, could be pushed further to show that unrestricted resolution must be exponential for Tseitin' examples. This would settle questions 2) and 5), but not necessarily question 4). It should be noted that for small examples, introducing nonregular derivations does not shorten the proofs. However as in most combinatorial problems, small examples do not mean anything and large examples are too complicated to analyze completely. This was exactly the difficulty we encountered when we tried to answer the questions about the bounded resolution versions. The open problem which seems the hardest is problem 1). Since extended resolution is so powerful, we feel that question 1) will be settled only when the two open problems involving P and NP are solved.

A final open problem is,

6) are there infinitely many $n \geq 1$ such that $\hat{f}_n \geq cn$ for some constant c? (where $\hat{f}_n$ is as defined in Section 7)

A positive answer to 6) would strengthen the results somewhat. It would imply a 2^{cn} lower bound for regular resolution and a cn lower bound for the size of clauses on a resolution tree. Answering 6) positively would require the construction of a family of graphs $\{G_n\}$ such that G_n achieves $\hat{f}_n$.

Acknowledgments

I wish to thank Professors J. Hopcroft and J. Hartmanis for their encouragement, advice and suggestions. Professor Hopcroft has also suggested BR1 of Section 5. Thanks also to Dr. J. Simon for many discussions, and to Dr. P. van Emde Boas and M. Solomon for help with the presentation.

References

[1] Cook, S.A. The complexity of theorem proving procedures. Proc. 3rd ACM STOC, 1971, 151-158.

[2] Cook, S.A. Examples for the Davis-Putnam procedure. Unpublished manuscript, June 1971.

[3] Cook, S.A. and R. Reckow. On the length of proofs in the propositional calculus. Proc. 6th ACM STOC, 1974, 135-148.

[4] Davis, M. and H. Putnam. A computing procedure for quantification theory. JACM 1, (1960), 201-215.

[5] Dunham, B. and H. Wang. Toward feasible solutions of the tautolog problem. IBM Thomas J. Watson Research Center RC 4924, July 1974

[6] Harris, R.V. A polynomial bound on the complexity of the Davis-Putnam procedure as applied to symmetrizable propositions. Ph.D. Thesis, Dept. of Computer Science, Cornell University, August 1972.

[7] Kirkpatrick, D. -- personal communication.

[8] Quine, W.V. A way to simplify truth functions. American Mathemati Monthly 62 (1956), 627-631.

[9] Rabin, M.O. Theoretical impediments to artificial intelligence. Proc. IFIP Mathematical Aspects of Information Processing, 1974, 615-619.

[10] Robinson, J.A. A machine oriented logic based on the resolution principle. JACM 12 (1965), 23-41.

[11] Simon, I. On the time required by the Davis-Putnam tautology recognition algorithm. Research Report CSRR-2050, Dept. of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1971.

[12] Simon, J. -- personal communication.

[13] Tseitin, G.S. On the complexity of derivations in the propositional calculus. Structures in Constructive Mathematical Logic, Part II. A.O. Silenko, (ed.), 1968, 115-125.