Notre Dame Journal of Formal Logic
Volume 27, Number 4, October 1986

Gentzen Systems, Resolution, and Literal Trees

DANIEL J. DOUGHERTY*

1 Introduction We are concerned with the relationship between various proof systems for propositional logic, with particular emphasis on the size of derivations. Two traditional systems, Gentzen's sequent calculus, and Robinson's resolution method, are investigated by introducing a (somewhat) new formalism, which may be viewed as a common generalization.

Cook and Reckhow, in [4] and [5], studied many logical calculi, including resolution and Gentzen systems, viewing these as nondeterministic algorithms, and reported polynomial time simulation results among certain systems. Both [4] and [5] contain discussions of the connection with computational complexity. More recently, Haken, in his thesis [8], showed that resolution is not a polynomially bounded system. Our emphasis here is on the structure of the derivations themselves, and upon obtaining inference-by-inference transformations, usually preserving the relation of subderivation. Our notation is as follows:

Propositional Logic: We assume an infinite set of literals p_i; these come in complementary pairs, the complement of p is denoted $\bar{p}$. *Formulas* are defined as follows: a literal is a formula, and if $A_1,\ldots,A_k$ are distinct formulas, then $\wedge\{A_1,\ldots,A_k\}$ and $\vee\{A_1,\ldots,A_k\}$ are formulas. We sometimes write $A_1 \wedge \ldots \wedge A_k$ and $A_1 \vee \ldots \vee A_k$. A disjunction of literals $\vee\{p_1,\ldots,p_k\}$ is a *clause*; it is convenient to refer to a clause by juxtaposing its literals: $p_1 p_2 \ldots p_k$.

The *negation* $\sim A$ of a formula A is defined by induction: if A is a literal, then $\sim A$ is $\bar{A}$; if A is $\wedge\{A_1,\ldots,A_k\}$, then $\sim A$ is $\vee\{\sim A_1,\ldots,\sim A_k\}$; if A is $\vee\{A_1,\ldots,A_k\}$, then $\sim$A is $\wedge\{\sim A_1,\ldots,\sim A_k\}$.

All of the systems we consider are refutation systems, that is, they demonstrate unsatisfiability, usually of sets of clauses. We will, however, use the terms proof, refutation, and derivation synonymously to refer to objects in these systems.

Trees: Our trees are finite, rooted trees, which we draw branching downward. If N^- is on the path from the root to N^+, $N^- \neq N^+$, we say N^+ lies *below* N^-, and write $N^- < N^+$. If N^+ lies immediately below N^-, we say that

*This is a portion of the author's Ph.D. dissertation, written under the direction of Prof. James C. Owings, at the University of Maryland. Partial support was provided by NSF Grant MCS-7919418.

Received May 14, 1984; revised March 22, 1985

N^+ is a *child* of N^-. If N_1 and N_2 are children of the same node, we say they are *adjacent*. If N is any node, $adj(N) = \{N' : Np \text{ is adjacent to } N\}$. Note that $N \in adj(N)$.

A *leaf* node is a node with no children; an *interior* node is a nonleaf node.

We will make constant use of the following conventions when discussing nodes. The letter K will name "conjunctive" nodes, these will have exactly one child K_1. The letter D will stand for a "disjunctive" node; with children $D_1, \ldots, D_k$, $k > 1$. Certain nodes with precisely two children will be "cut-nodes"; the letter X will always denote one of these, with children X_1 and X_2. Leaf nodes will be called L. Arbitrary nodes will be designated by N or M.

A *path* is a sequence of nodes $\langle N_1, \ldots, N_k \rangle$ such that N_{i+1} is a child of N_i, $1 \le i \le k - 1$. A *branch* is a path with N_1 = root, and N_k a leaf node. If N is any node, the *subtree generated by N* is the subgraph determined by N and all the nodes below N.

We will usually be considering pairs of the form (T, h), where h is a function defined on nodes of T, sometimes just the nonroot nodes. In this case, we define $Sub(N)$ to be the pair (T', h'), where T' is the subtree generated by N, and h' is the restriction of h to T', with the understanding that if h was not defined for the root of T, then h' is not defined for N. Thus $Sub(N)$ is an object of the same type as (T, h).

2 Literal trees and resolution

2.1 Definition A *literal tree* is a pair $\mathfrak{I} = (T, l)$, where $\mathfrak{I}$ is a tree and l maps nonroot nodes of T to literals. If X is a node with precisely two children, X_1 *and* X_2, with $l(X_1) = \overline{l(X_2)}$, then X is a *cut* node. If N is not a cut node and $N_1, \ldots, N_k$ are the children of N, then the clause $\vee\{l(N_1), \ldots, l(N_k)\}$ *occurs* on $\mathfrak{I}$. If $\mathcal{S}$ is the collection of clauses occurring on $\mathfrak{I}$, then $\mathfrak{I}$ is a literal tree *for* $\mathcal{S}$. If Π is a branch of $\mathfrak{I}$, ending in L, then Π is *closed* if for some N on Π, $l(N) = \overline{l(L)}$. $\mathfrak{I}$ is *closed* if each of its branches closes.

A branch of a literal tree may have more than one pair of complementary literals—we only insist that the leaf node find its literal complemented above it.

We will occasionally speak of "the node p" when we mean a particular node N that $l(N) = p$. However, since l need not be one-to-one, we usually must be careful to distinguish between N and $l(N)$.

The following characterization of the set of clauses occurring on $\mathfrak{I}$ is convenient. We simultaneously define the set of clauses $Occ(N)$, for each node N:

(1) $Occ(L) = l(L)$
(2) $Occ(X) = Occ(X_1) \cup Occ(X_2)$
(3) $Occ(D) = \bigcup \{Occ(D_i)\} \cup \{\vee\{l(D_i)\}\}$.

Then $Occ(N)$ is the set of clauses occurring on $Sub(N)$, and we define $Occ(\mathfrak{I})$ to be $Occ(\text{root of } T)$.

2.2 Theorem *Let S be a set of clauses. Then S is unsatisfiable iff there exists a closed literal tree for a subset of S.*

Proof: Suppose S is unsatisfiable. The full binary tree of truth assignments to the atoms in S is such that for each branch of the tree there is a clause in S, all of whose literals are complemented in the branch. Thus, we can build a closed tree by appending a suitable clause to the end of each branch.

If S is satisfiable, and $\mathfrak{I}$ is any literal tree for S, we can construct an open branch through $\mathfrak{I}$ by choosing a literal from the successful truth assignment for S at each level.

2.3 Definition Let $\mathfrak{I} = (T, l)$ be a literal tree, and let M, N be nodes of T. If the subliteral trees generated by M and N are isomorphic, we say that M and N are *equivalent*, and write $M \simeq N$.

Thus, $M \simeq N$ if the patterns of nodes and the associated literals below M and N are the same, but it is *not* required that $l(M) = l(N)$ in $\mathfrak{I}$. Since literal trees do not assign literals to their roots, $l(N)$ is irrelevant to the subliteral tree generated by N.

The above relation $\simeq$ is clearly an equivalence relation.

2.4 Definition If $\mathfrak{I} = (T, l)$ is a literal tree,
(i) $|\mathfrak{I}|$ = the number of interior nodes of T.
(ii) $\#\mathfrak{I}$ = the number of $\simeq$ equivalence classes of interior nodes of T.

We have chosen to count only the interior nodes of trees, to facilitate comparison with other systems. The total number of nodes in a tree is then bounded by $n|\mathfrak{I}|$, where n is the length of the longest clause in S. If one has a complexity-theoretic interest in proof systems as nondeterministic algorithms, it is worth recalling here that any set of clauses can be transformed into a set with three literals per clause, with a linear change in the length of the set, without affecting satisfiability.

2.5 Definition
(i) An *analytic* literal tree is a literal tree with no cut nodes.
(ii) A *cut tree* is a literal tree such that for all interior nodes N, either N is a cut node, or all of N's children are leaves.
(iii) A *regular* literal tree is one for which, in any path, $N_1 \neq N_2$ implies $l(N_1) \neq l(N_2)$.

The tree constructed in the proof of Theorem 2.2 is a regular cut tree. In general, a branch Π in a literal tree for S may be regarded as a potential (partial) truth assignment for S. If the leaf-literal of Π finds its complement above it, then Π is abandoned as a candidate for satisfying S. An irregular tree $\mathfrak{I}$ is simply one in which certain branches contain redundant occurrences of literals. An irregular *cut* tree $\mathfrak{I}$ can actually have branches which contain complementary pairs of literals (recall that a branch is closed only if its *leaf* literal is complemented above; if we are building a tree, we do not necessarily terminate a branch when we come to a literal whose complement appears above). For example, if $\mathfrak{I}$ is a cut tree with a branch containing the literal p twice, then the lower occurrence of p will be adjacent to a $\bar{p}$, which is indeed a descendant of p, and is not, in general, a leaf-literal. In this case, one of the branches of $\mathfrak{I}$ corresponds to a contradictory "truth assignment".

In fact, it will be shown later in this chapter that "irregularities" of this type can be removed from a tree $\mathfrak{T}$ without increasing $|\mathfrak{T}|$, but there is evidence that regular trees are *not* minimal with respect to #-size. At present, there is no known method of removing irregularities without increasing the #-size. This is discussed further below.

As an example, let $\mathfrak{C}$ be the set of clauses $\{\bar{p}, pq, r\bar{x}, \bar{q}x, \bar{q}\bar{r}\}$ and let $\mathfrak{T}$ be the closed literal tree for $\mathfrak{C}$ shown in Figure 1.

The node labeled q is a cut node, and the two nodes labeled $\bar{x}$ are equivalent. The tree is not regular, because of the branch with labels q, x, $\bar{x}$, x. We have $|\mathfrak{T}| = 7$, $\#\mathfrak{T} = 6$.

A cut node in a literal tree is reminiscent of a cut-inference in a Gentzen-style derivation. (The correspondence between literal trees and Gentzen systems is explored fully in Section 3.) A cut-elimination theorem for literal trees is obtained as follows.

2.6 Theorem *If a set of clauses S is unsatisfiable, then there exists an analytic closed literal tree for S.*

Proof: Let $\mathfrak{T}$ be a closed literal tree for S. The proof is by induction on the number of $\simeq$ equivalence classes of cut nodes in $\mathfrak{T}$; we give a procedure for eliminating these as follows:

Let X be a cut node such that no $X' \simeq X$ has a cut node above it; say that $l(X_1) = p$, $l(X_2) = \bar{p}$, and let $\mathfrak{T}_1$ be the subtree generated by p, $\mathfrak{T}_2$ generated by $\bar{p}$.

Form the new tree $\mathfrak{T}'$ as follows. If L is a leaf node of $\mathfrak{T}$ which is below X_2, with $l(L) = p$, call L an orphan. Now delete X_1 and X_2, letting the new children of X be the old children of X_2, and attach a copy of $\mathfrak{T}_1$ below each

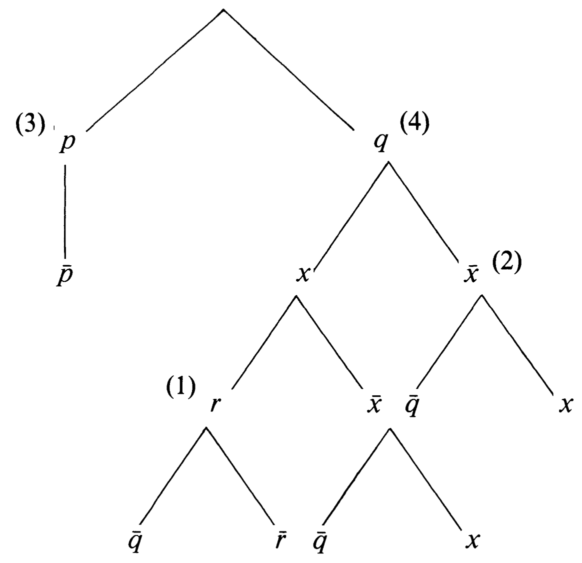


Figure 1

orphan. For any $X' \simeq X$, perform the same procedure. We have eliminated all occurrences of nodes equivalent to X, and have not introduced any new $\#\sim$classes of cut nodes (by the choice of X, if two cut nodes were $\sim$ in $\mathfrak{I}$, they are $\sim$ in $\mathfrak{I}'$). It remains to show that $\mathfrak{I}'$ is closed.

Let L be a leaf node of $\mathfrak{I}'$. If L was a leaf node of $\mathfrak{I}$ originally, then L is not one of the L_i, thus $l(L)$ is not p. But the only change in the set of literals above L is the deletion of the $\bar{p}$ node. Thus $\overline{l(L)}$ is still above L in $\mathfrak{I}'$, and the branch ending in L closes. If L was a leaf node from $\mathfrak{I}_1$ in $\mathfrak{I}$, then the set of literals above L in $\mathfrak{I}'$ contains those above L in $\mathfrak{I}$ (recall that copies of $\mathfrak{I}_1$ are attached below occurrences of p), so again $\overline{l(L)}$ lies above L in $\mathfrak{I}'$.

In general, the construction above will increase both the $|\cdot|$ and #-measure of tree size. It will follow from later results that cut-trees are actually the trees of minimal $|\cdot|$ and #-size (to within a factor of 2). The remarks following Corollary 3.26 contain the implications of this for Gentzen system derivations.

Analytic literal trees are, of course, special sorts of analytic tableaux, which were devised independently by Beth [1], Hintikka [9], and Schutte [14] and were developed and streamlined by Smullyan [15].

2.7 Definition The *resolution rule* is the following rule of inference: From clauses $p \cup C_1$ and $\bar{p} \cup C_2$, infer $C_1 \cup C_2$. The clause $C_1 \cup C_2$ is the *resolvent* of $p \cup C_1$ and $\bar{p} \cup C_2$, and p is the literal *resolved upon*. If both C_1 and C_2 are empty, then the resolvent (of p and $\bar{p}$) is the empty clause $\Box$.

Let S be a set of clauses. A *resolution refutation* of S is a pair $\mathcal{R} = (T, c)$, where T is a binary tree and c maps nodes of T to clauses, such that

(i) $\{c(L): L \text{ is a leaf of } T\} \subseteq S$,
(ii) $c(root) = \Box$,
(iii) if N is the parent of N_1 and N_2, then $c(N)$ is a resolvent of $c(N_1)$ and $c(N_2)$,
(iv) if $c(N) = c(N')$, then the subderivations generated by N and N' are isomorphic.

We will also refer to such an $\mathcal{R}$ as a resolution *proof* of S.

A proof procedure based on resolution was introduced by Robinson [12] as a means of verifying the unsatisfiability of sets of clauses in the predicate calculus, and as such is the basis of much automatic theorem proving (see, for example, [3]).

2.8 Theorem *Let S be a set of clauses. Then S is unsatisfiable iff there exists a resolution refutation of S.*

Proof: See, e.g., [3]. An alternate proof is implicit in Theorems 2.10 and 2.12 below.

2.9 Definition Let $\mathcal{R} = (T, c)$ be a resolution proof.

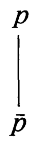
(i) $|\mathcal{R}|$ = the number of interior nodes of T. This is the number of inferences in $\mathcal{R}$.
(ii) $\#\mathcal{R}$ = the number of distinct clauses appearing among the interior nodes. This equals the number of *distinct* inferences in the proof.

Figure 2 gives an example of a resolution proof.

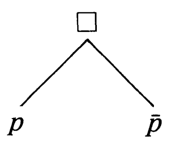
2.10 Theorem *Let $\mathfrak{T} = (T, l)$ be a closed literal tree for S. Then there exists a resolution refutation $\mathfrak{R} = Res(\mathfrak{T})$, of S with $|\mathfrak{R}| \leq |\mathfrak{T}|$ and $\#\mathfrak{R} \leq \#\mathfrak{T}$.*

Proof: The proof is by induction on $\#\mathfrak{T}$.

When $\#\mathfrak{T} = 1$, then $\mathfrak{T}$ is of the form:



Thus $|\mathfrak{T}| = 1$, S is $\{p, \bar{p}\}$, and we can let $\mathfrak{R} = Res(\mathfrak{T})$ be the following proof, with $|\mathfrak{R}| = 1$ and $\#\mathfrak{R} = 1$:



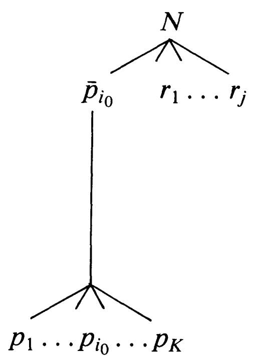
Now, suppose $\#\mathfrak{T} > 1$, and let S be the set of clauses occurring on $\mathfrak{T}$. We construct a closed literal tree $\mathfrak{T}_1$ for a set of clauses S_1, such that

(i) $|\mathfrak{T}_1| < |\mathfrak{T}|$, $\#\mathfrak{T}_1 < \#\mathfrak{T}$,

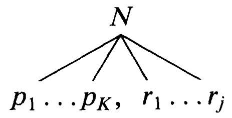
(ii) $S_1 = S$ or S_1 is obtained from S by one resolution inference.

This will establish, by induction, the existence of a resolution proof $\mathfrak{R}$ of S, with $|\mathfrak{R}| < |\mathfrak{T}|$ and $\#\mathfrak{R} < \#\mathfrak{T}$.

Let p be any leaf node such that its adjacent nodes are also leaves. Let $C = p_1 p_2 \ldots p_k$ be the clause in S consisting of p and its adjacent literals. Now let $\bar{p}_{i_0}$ be the *lowest* node in the path through p which is complementary to one of the p_i, and let $c^- = \{\bar{p}_{i_0}, r_1, \ldots, r_j\}$ be the clause given by $\bar{p}_{i_0}$ and its adjacent nodes. Let N be the parent node of the literals in c^-:



To construct $\mathfrak{Z}_1$, delete the node $\bar{p}_{i_0}$ and the subtree below it, and add new children of N corresponding to each p_i from C which is not p_{i_0} nor is equal to any of the r_i in C^-. If N' is any other node of $\mathfrak{Z}$ with $N \simeq N'$, perform the same operation below N'.



Since the node p_{i_0} is annihilated, $|\mathfrak{Z}_1| < |\mathfrak{Z}|$. Since all $N' \simeq N$ were treated alike, $\#\mathfrak{Z}_1 < \#\mathfrak{Z}$.

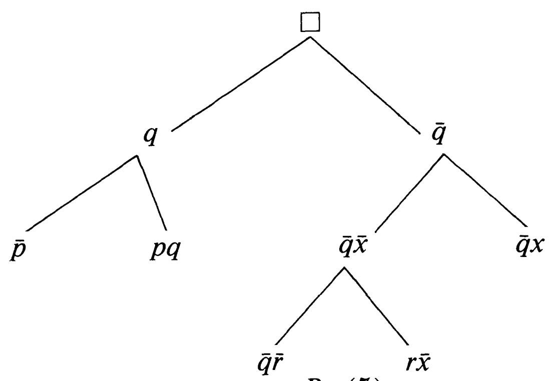
If N was a cut node of $\mathfrak{Z}$, then the new clause below N is simply C, thus $S_1 = S$. If N was not a cut node, then C and C^- are clauses from S, and thus S_1 is obtained from $\mathcal{C}$ by resolution.

It remains to see that $\mathfrak{Z}_1$ is closed. The only branches of $\mathfrak{Z}_1$ which are not identical with branches of $\mathfrak{Z}$ are those ending in one of the p_i from C. Each of these p_i were complemented in $\mathfrak{Z}$, and since $\bar{p}_{i_0}$ was the *lowest* $\bar{p}_i$ occurring as such a complement, the remaining $\bar{p}_i$ still serve to close the respective branches ending in p_i.

2.11 Corollary *If $\mathfrak{Z}$ is a closed cut tree for S, then there exists a resolution proof $\mathcal{R}$ of S, with $|\mathcal{R}| \leq \frac{1}{2}|\mathfrak{Z}|$.*

Proof: In the construction of $Res(\mathfrak{Z})$, when the node N is a cut node we actually eliminate both children of N with one resolution (the first new clause below N is identical with the clause being brought up from below). If all interior nodes of $\mathfrak{Z}$ are cut nodes, then we can associate with each inference of $Res(\mathfrak{Z})$ a *pair* of interior nodes of $\mathfrak{Z}$, namely the complementary pair of cut-literals below the node N in the proof. Hence $|\mathcal{R}| \leq \frac{1}{2}|\mathfrak{Z}|$.

In Figure 2, we illustrate the resolution proof $Res(\mathfrak{Z})$ derived from the



$Res(\mathfrak{Z})$

Figure 2

literal tree $\mathfrak{I}$ of Figure 1. The numbers of the nodes in $\mathfrak{I}$ correspond to inferences in $Res(\mathfrak{I})$.

2.12 Theorem *Let $\mathfrak{R}$ be a resolution proof of S. Then there exists a closed cut tree $\mathfrak{I} = Tr(\mathfrak{R})$ for S, with $|\mathfrak{I}| = 2|\mathfrak{R}| + 1$ and $\#\mathfrak{I} = \#\mathfrak{R} + \#S$, where $\#S$ is the number of clauses in S.*

Proof: Let $\mathfrak{R} = (T, c)$, and let L be a leaf node of T, so that $c(L) = p_1 \ldots p_k$ is a clause of S. For each such L, add k children $L_1, \ldots, L_k$ to L, which will serve as leaves of a new tree T'. The literal tree $\mathfrak{I}$ will be (T', l) where l is defined as follows. For a node L_i as above, define $l(L_i) = p_i$. If N is an interior node of T, then its children N_1 and N_2 are interior nodes of T', and $c(N_1)$ and $c(N_2)$ resolve to give $c(N)$. If $|p|$ is the literal resolved upon, with $p \in c(N_1)$ and $\bar{p} \in c(N_2)$, we define $l(N_1) = \bar{p}$ and $l(N_2) = p$. It is easy to check that $\mathfrak{I}$ is the desired tree.

Note that although $Tr(Res(\mathfrak{I}))$ need not equal $\mathfrak{I}$, $Res(Tr(\mathfrak{R}))$ will always be $\mathfrak{R}$, so that after the first iteration, the operations Tr and Res are inverses of each other, and the factor of 2 in tree size is introduced only once.

The elimination of cuts from a literal tree can lead to an increase in proof size (we will say more on this in Section 3). We can observe now, though, that cut-trees are in fact minimal in size, up to a factor of 2:

2.13 Corollary *Given any closed literal tree $\mathfrak{I}$ for S, there is a closed cut tree $\mathfrak{I}'$ for S, with $|\mathfrak{I}'| \leq 2|\mathfrak{I}| + 1$ and $\#\mathfrak{I}' \leq \#\mathfrak{I} + \#S$.*

Proof: Let $\mathfrak{I}' = Tr(Res(\mathfrak{I}))$.

Semantic trees, defined in Kowalski and Hayes [11], correspond to certain regular cut trees, in fact, those of the type constructed in Theorem 2.2. These were introduced to study resolution refutations, but Theorem 2.12 fails if "cut tree" is replaced by "semantic tree".

A significant question to be raised concerning a propositional proof system is, "What is the relationship between the length of a provable formula and the length of its shortest proof in the given system?" The following theorem is due to Haken [8]:

2.14 Theorem *There exists $c > 0$ and unsatisfiable sets of clauses S_n of size $O(n^3)$, such that for infinitely many n, every resolution refutation $\mathfrak{R}$ of S_n has $\#\mathfrak{R} \geq 2^{cn}$.*

The notion of extended resolution was introduced by Tseitin in [16], in his attempt to prove an exponential lower bound for the complexity of resolution. He was able to define a class of sets of clauses which seemed to require long resolution refutations, but in fact were not intrinsically difficult to refute, if auxiliary literals were introduced.

If S is a set of formulas, the *extension* rule may be applied to yield a new set S^*. Let $A_1, \ldots, A_k$ be formulas, and let x be a literal not occurring in any formula of S. Then any occurrence $\vee\{A_1, \ldots, A_k\}$ in S may be replaced by x, and any occurrence of $\wedge\{\sim A_1, \ldots, \sim A_k\}$ may be replaced by $\bar{x}$, provided the following set of clauses is added: $\vee\{\bar{x}, A_1, \ldots, A_k\}$, $\vee\{x, \sim A_1\}, \ldots, \vee\{x,$

$\sim A_k\}$. The conjunction of the new clauses added to S is logically equivalent to $x \leftrightarrow \vee\{A_1,\ldots,A_k\}$.

2.15 Definition An *Extended Resolution* refutation of S is a resolution refutation of a set S^*, where S^* is obtained from S by finitely many applications of extension.

2.16 Theorem (Cook and Reckhow) *For each of the sets S_n in Theorem 2.14, there is an Extended Resolution refutation $\mathcal{E}$, with $\#\mathcal{E} = O(n^4)$.*

Extended Resolution has been investigated by Cook and Reckhow [5], who prove that this system is powerful enough to polynomially simulate almost all standard proof systems for propositional logic, including Natural Deduction, Hilbert-style system, and Gentzen-style systems. Cook's paper [2] discusses the relationship of Extended Resolution to Cobham's class $\mathcal{L}$ (the class of polynomial-time computable functions). Section 3 clarifies the connection between Extended Resolution and the sequent calculus.

2.17 Definition A resolution proof $\mathcal{R} = (T, c)$ is *regular* if no path in T contains two clauses which are the results of resolutions on the same atom.

Regular Resolution, also due to Tseitin, is the proof system obtained from Resolution by requiring proofs to be regular.

It is clear from Theorem 2.12 that $\mathcal{R}$ is a regular resolution proof iff $Tr(\mathcal{R})$ is a regular literal tree. Tseitin was able to show that Regular Resolution, as a nondeterministic algorithm, is exponentially complex, and Galil improved his lower bound [16], [6]. Tseitin observed that Regular Resolution was minimal with respect to the tree size of proofs.

2.18 Theorem (Tseitin) *For every Resolution refutation $\mathcal{R}$ of S, there is a Regular Resolution $\mathcal{R}'$ of S, with $|\mathcal{R}'| \le |\mathcal{R}|$.*

It is not known whether Regular resolution is #-minimal. This is equivalent to the question of whether regular cut-trees are minimal with respect to #-size, that is (cf. the discussion following Definition 2.5), whether or not trees with contradictory truth assignments on their branches allow for #-shorter proofs.

Tseitin did not provide a proof in [16] of Theorem 2.18, but Galil gives a proof in [7]. It is easy to prove Theorem 2.18 by passing to literal trees, and we leave this to the interested reader. In fact, the technique is similar to that used in the following lemma, to the effect that trees need not contain nodes which are not used to close branches. This can lead to a considerable decrease in the size of trees.

2.19 Lemma *Let $\mathfrak{J}_0$ be a closed literal tree for S. Then there exists a closed literal tree $\mathfrak{J}_1$, for a subset of S, with $|\mathfrak{J}_1| \le |\mathfrak{J}_0|$ and $\#\mathfrak{J}_1 \le \#\mathfrak{J}_0$, such that for every nonroot interior node N, there exists a leaf L below N, with $l(N) = \overline{l(L)}$, and N is the lowest node complementing L.*

Proof: Let N be a node of $\mathfrak{J}_0$, with parent M and children $N_1,\ldots,N_k$, not satisfying the conclusion of the lemma. Let $\mathfrak{J}'$ be obtained by deleting N and its adjacent nodes, that is, by letting the new children of M be the $N_1,\ldots,N_k$. If we perform the same operation below each $M' \simeq M$, then $|\mathfrak{I}'| \leq |\mathfrak{I}|$ and $\#\mathfrak{I}' \leq \#\mathfrak{I}$, and iterating this procedure will eventually yield a tree $\mathfrak{I}_1$ satisfying the lemma. We show that $\mathfrak{I}'$ is closed as follows:

The only leaves whose branches have been altered are those below some $M' \simeq M$; these have had an occurrence of $l(N)$ deleted. But N (and any $N' \simeq N$) has the property that any branch passing through it with a node labeled $\overline{l(N)}$ must have a node lower than N, labeled $l(N)$. Thus the deletion of N will not affect the closing of the branch.

3 Gentzen systems

In this section we examine the relationship between resolution-based proof systems and Gentzen-style systems, and explore the consequences of restricting the form of inferences allowed in the sequent calculus.

3.1 Definition A *sequent* Δ is a finite set of formulas. A sequent $\{A_1, \ldots, A_k\}$ is said to be *unsatisfiable* if the formula $\wedge\{A_1, \ldots, A_k\}$ is unsatisfiable. As usual, we write Δ, A in place of $\Delta \cup \{A\}$ and Δ_1, Δ_2 in place of $\Delta_1 \cup \Delta_2$. A sequent of the form $\Delta, A, \sim A$ is an *axiom*. *Rules* are configurations of sequents:

$$(\vee) \qquad \frac{\Delta_1, \ldots, \Delta_n, A_1 \vee A_2 \vee \ldots \vee A_n}{\Delta_1, A_1 | \Delta_2, A_2 | \ldots | \Delta_n, A_n}$$

$$(\wedge) \qquad \frac{\Delta, A_1 \wedge \ldots \wedge A_n}{\Delta, A_1, \ldots, A_n}$$

$$(\text{cut}) \qquad \frac{\Delta_1, \Delta_2}{\Delta_1, A | \Delta_2, \sim A}.$$

Note that we have written our rules upside-down in comparison with standard treatments—this is to facilitate the comparison with literal trees and resolution.

In each rule, the sequent on top is the *derived* sequent, the sequents below are *premises*. In each premise, the formulas A_i are the *minor formulas* (m.f.). In the ($\wedge$) rule, $A_1 \wedge \ldots \wedge A_n$ is the *principal formula* (p.f.), in the ($\vee$) rule, $A_1 \vee \ldots \vee A_n$ is the p.f. Cut has no p.f.

We will write m.f. (N) to denote the set of minor formulas of $seq(N)$.

3.2 Definition Let G_l be the collection of rules above, with the restriction that in each inference rule and axiom, the A_i must be literals.

Let G be the collection of rules with no restrictions.

G_l is thus the proof system which can only manipulate literals; conjunctions and disjunctions must be formed at one step, and are not eligible to enter into subsequent inferences. It is this feature which makes G_l the precise counterpart to Resolution (cf. Corollaries 3.14 and 3.16 below). Resolution, of course, treats a clause $(A_1 \vee \ldots \vee A_k)$ as a unit and does not allow a resolution to be performed on, say $(A_1 \vee A_2)$. The flexibility, in system G, to treat a disjunction $(A_1 \vee A_2)$ in different ways, for example as part of a disjunction $((A_1 \vee A_2) \vee B)$ or a conjunction $((A_1 \vee A_2) \wedge C)$, is the analogue of the extension rule (cf. Corollary 3.22). A sequent containing $A_1 \vee A_2$ can be derived in G then used in different places in a proof, allowing for shorter proofs (cf. Corollary 3.25).

Note that system G allows axioms of the form Δ, A, $\sim A$, where A is not necessarily a literal. It is easy to see that such sequents are provable in a system allowing only axioms Δ, p, $\sim p$, by proofs linear in the length of A, and the general axioms are more in the spirit of system G.

3.3 Definition Let Δ be a sequent. A *Gentzen derivation* $\mathcal{G}$ of Δ in system $G[G_l]$ is a pair (T, seq), in which T is a tree and seq is a function mapping nodes of T to sequents such that

(i) $seq(root) = \Delta$,
(ii) if L is a leaf of T, $seq(L)$ is an axiom of $G[G_l]$,
(iii) if N is an interior node, with children $N_1, \ldots, N_k$, then

$$\frac{seq(N)}{seq(N_1) | \ldots | seq(N_k)}$$

is in the form of a rule of inference of $G[G_l]$, and
(iv) if $seq(N) = seq(M)$, then the subderivations below N and M are isomorphic.

Provision (iv) of Definition 3.3 is not necessary to the completeness or soundness of the proof systems, but since it clearly does not alter the set of provable sequents, and the *shortest* proofs of a given sequent will always satisfy (iv), we find it convenient to insist on this property.

We may characterize derivations in another way, as follows. A derivation is a tree T labeled with sequents, each of which has a specified set of minor formulas, such that:

(i) $seq(L)$ is an axiom,
(ii) $seq(K) = [seq(K_1), \wedge\{\text{m.f.}(K_1)\}] - \{\text{m.f.}(K_1)\}$,
(iii) $seq(D) = [seq(D_i), \vee\, \text{m.f.}(D_i)] - \bigcup \text{m.f.}(D_i)$,
(iv) $seq(X) = [seq(X_1), seq(X_2)] = [\text{m.f.}(X_1), \text{m.f.}(X_2)]$.

3.4 Definition Let $\mathcal{G} = (T, seq)$ be a Gentzen proof, let N be a node of T.
(i) The weight of N, $wt(N)$, is the sum of the number of minor formulas used in deriving p.f.(N).
(ii) $|\mathcal{G}| = \Sigma wt(N)$, the sum taken over all nodes of T.
(iii) $\#\mathcal{G}$ = the number of distinct nonaxiom sequents occurring in G.

Note that if $\mathcal{G}$ is a G_l proof of a set of clauses S, then $wt(N) = 1$ for each node N, so that $|\mathcal{G}|$ is simply the number of interior nodes in the underlying tree, that is, the number of nonaxiom sequent occurrences in $\mathcal{G}$.

We note here that if a standard Gentzen system, such as in [10], is transformed into a "left-handed" one, and our convention of automatically putting formulas in negation-normal form is implemented, then we obtain a line-by-line translation of a standard proof into a G-proof. On the other hand, our treatment of $\vee$ and $\wedge$ as operations on sets rather than as binary relations is only a convenience, and has no significant complexity ramifications: any inference with p.f. $A_1 \vee \ldots \vee A_k$ or $A_1 \wedge \ldots \wedge A_k$ can be replaced by $k - 1$ inferences in a system treating $\vee$ and $\wedge$ as binary operations. To summarize:

3.5 Proposition *Given any standard proof P of a standard sequent $A_1,\ldots,A_n \rightarrow B_1,\ldots,B_k$, there exists a proof $\mathcal{G}$ of the sequent $A_1,\ldots,A_n$, $B_1,\ldots,B_k$ in system G, with $|\mathcal{G}| \leq$ the number of sequent-occurrences in P and $\#\mathcal{G} \leq$ the number of distinct sequents in P.*

Proof: We outlined above a method of transforming P into $\mathcal{G}$.

3.6 Corollary *A sequent Δ is unsatisfiable iff there is a derivation in system G of Δ.*

System G_l is sound, and is complete for unsatisfiable sets of clauses. This is implied by Theorems 3.10 and 3.13 below.

We collect here three technical lemmas which will be used below.

3.7 Lemma *In any G proof, if a formula A appears in some seq(N), then it appears in seq(N^+), for all $N^+ \geq N$, unless $A \in$ m.f. N' for some N' in the path between N^+ and N.*

Proof: This is immediate from the characterization of *seq*(M) given following Definition 3.3.

3.8 Lemma *Given any G proof $\mathcal{G}$ of Δ, there is a proof $\mathcal{G}'$ of Δ, $|\mathcal{G}'| \leq |\mathcal{G}|$, $\#\mathcal{G}' \leq \#\mathcal{G}$, such that for each axiom Γ, A, $\sim A$ of $\mathcal{G}'$, either A or $\sim A$ is a minor formula of that axiom.*

Proof: Suppose L is a leaf node of $\mathcal{G}$, with *seq*$(L) = \Gamma, A, \sim A$, m.f. $(L) \subseteq \Gamma$. Then by Lemma 3.7, both A and $\sim A$ are in *seq*(L^*), where L^* is the parent of L, thus *seq*(L^*) is a G-axiom. We can thus delete L from $\mathcal{G}$, and continue this process until one of A or $\sim A$ is used as a minor formula.

Next, we note that instances of $\wedge$-rule, since it involves no branching, can be "pushed upward" toward the root of the tree, with no increase in proof size.

3.9 Lemma *Let $\mathcal{G}$ be a G-proof of Δ. Then there is a proof $\mathcal{G}'$ of Δ, $|\mathcal{G}'| \leq |\mathcal{G}|$, $\#\mathcal{G} \leq \#\mathcal{G}'$,* such that, if *$\wedge\{A_1,\ldots,A_k\} \in m.f.(N)$ in $\mathcal{G}'$, N has precisely one child N_1, with $m.f.(N_1) = \{A_1,\ldots,A_k\}$.*

Proof: Let Γ, $(A_1 \wedge \ldots \wedge A_k)$ be the lowest sequent below N containing $(A_1 \wedge \ldots \wedge A_k)$.

Construct $\mathcal{G}_1$ by first adding the node Δ', $A_1,\ldots,A_n$ just below N, and deleting all lower inferences in which $(A_1 \wedge \ldots \wedge A_n)$ is a p.f.; then repeating this process beneath every other occurrence of Δ', $(A_1 \wedge \ldots \wedge A_n)$. We are calling upon part (iv) of Definition 3.3, of course.

$\mathcal{G}_1$ is still a G-proof, and the top-sequent has not changed. The new node (sequent) Δ', $A_1,\ldots,A_k$ has been paid for by the disappearance of Γ, $(A_1 \wedge \ldots \wedge A_k)$, thus $|\mathcal{G}_1| \leq |\mathcal{G}|$, $\#\mathcal{G}_1 \leq \#\mathcal{G}$.

Iterating this construction yields the desired $\mathcal{G}'$.

We can now show that system $\mathcal{G}_l$ corresponds precisely to Resolution, by exhibiting a correspondence between $\mathcal{G}_l$ and literal trees.

3.10 Theorem *Let S be a set of clauses, and let $\mathcal{G}$ be a G_l proof of S. Then there exists a closed literal tree $\mathfrak{I}$ for S, with $|\mathfrak{I}| \leq |\mathcal{G}| + m$, $\#\mathfrak{I} \leq \#\mathcal{G} + m$, where m is the number of single-literal clauses of S.*

Proof: Let $\mathcal{G} = (T, seq)$. Let $q_1, \ldots, q_m$ be the single-literal clauses which are elements of S. We now define $\mathfrak{I}$.

Replace the root of T (the underlying tree of $\mathcal{G}$) with a path of m nodes, labeled $q_1, \ldots, q_m$. We then let T' be formed by adding T below this configuration (that is, identify the root of T and the node underlying $q_m \ldots$). For nodes N of T' other than those underlying the q_i, we define $l(N)$ = m.f. of $seq(N)$. $\mathfrak{I} = (T', l)$ is then a literal tree. $|\mathfrak{I}| \leq |\mathcal{G}| + n$ is immediate, and $\#\mathfrak{I} \leq \#\mathcal{G} + n$ follows from this by the usual argument that identical sequents of $\mathcal{G}$ will lead to $\simeq$ nodes of $\mathfrak{I}$.

To see that $\mathfrak{I}$ is closed, let L be a leaf of T', hence a leaf of T, and suppose $l(L) = p$. Thus $seq(L)$ is $\Gamma, p, \bar{p}$. But by Lemma 3.7, $\bar{p}$ must occur as m.f. in the branch above L, or be one of the clauses in Δ. In either case, there is a node labeled with $\bar{p}$ in the $\mathfrak{I}$-branch above L, hence the branch is closed.

3.11 Corollary *Given any G_l proof $\mathcal{G}$ of S, there exists a resolution refutation $\mathcal{R}$ of S, with $|\mathcal{R}| \leq |\mathcal{G}| + \#S$, $\#\mathcal{R} \leq \#\mathcal{G} + \#S$.*

Proof: Immediate from Theorem 3.10 and Corollary 2.11.

3.12 Corollary *System G_l is not a polynomially bounded proof system.*

Proof: Immediate from Corollary 3.11 and Theorem 2.14.

We have a converse to Theorem 3.10:

3.13 Theorem *Let $\mathfrak{I}$ be a closed literal tree for S, a set of clauses. Then there exists a G_l proof $\mathcal{G}$ of S, with $|\mathcal{G}| = |\mathfrak{I}|$, $\#\mathcal{G} = \#\mathfrak{I}$.*

Proof: We assume that $\mathfrak{I} = (T, l)$ satisfies the conclusion of Lemma 2.19. $\mathcal{G}$ will have underlying tree T and seq defined by induction as follows:

$$\begin{aligned} seq(L) &= \{l(L), \overline{l(L)}\} \\ seq(X) &= [seq(X_1), seq(X_2)] - [l(X_1), l(X_2)] \\ seq(N) &= [\bigcup \{seq\, N_i\}, \vee\{l(N_i)\}] - \{l(N_i)\}, \end{aligned}$$

where $\{N_i : 1 \leq i \leq k\}$ are the children of N. $\mathcal{G}$ is clearly a G_l proof, and since $\mathcal{G}$ and $\mathfrak{I}$ have identical underlying trees, $|\mathcal{G}| \leq |\mathfrak{I}|$. Since the definition of $seq(M)$ above depends only on $Sub(M)$ in $\mathfrak{I}$, it follows that $M \simeq M'$ in $\mathfrak{I}$ implies $seq(M) = seq(M')$, hence $\#\mathcal{G} \leq \#\mathfrak{I}$.

It remains to show that $seq(root) = \Delta$. To do this, we prove by induction:

(*) $seq(N) = Occ(N) \cup \{p : \bar{p} = l(L)$ for some $L \leq N$ and $p \neq l(N^-)$ for all $N^- < N\}$.

This is immediate for leaf nodes. The definition of $seq(M)$ given above parallels the characterization of $Occ(M)$ given following Definition 2.1, except that $seq(L)$ contains both of p and $\bar{p}$, where $Occ(L)$ contains only p. Thus the extra $\bar{p}$ will persist in the sequents of the ancestors of L, until $\bar{p}$ is m.f. of a node N in $\mathcal{G}$. But this happens precisely when $\bar{p} = l(N)$ in $\mathfrak{I}$. This establishes *.

Now, since $\mathfrak{I}$ is closed, every L below the root has the complement of $l(L)$ above it, thus $seq(root) = Occ(root) = S$.

3.14 Corollary *For any resolution proof $\mathcal{R}$ of S, there is a G_l proof $\mathcal{G}$ of S, with $|\mathcal{G}| \leq 2|\mathcal{R}| + 1$, $\#\mathcal{G} \leq \#\mathcal{R} + \#S$, and such that every interior inference of $\mathcal{G}$ is an instance of Cut.*

Proof: Immediate from Theorems 2.12 and 3.13.

Cut elimination for G_l follows immediately from the version for literal trees, as does the minimality of cut proofs:

3.15 Corollary *Given a G_l proof $\mathcal{G}$ of S, one can construct a cut-free proof $\mathcal{G}'$ of S.*

Proof: From $\mathcal{G}$, obtain a literal tree $\mathfrak{I}$, then use Theorem 2.6 to eliminate cut-inferences from $\mathfrak{I}$, obtaining $\mathfrak{I}'$. Theorem 3.13 yields $\mathcal{G}'$.

3.16 Corollary *Let S be a set of clauses with m single-literal clauses. Given a G_l proof $\mathcal{G}$ of S, there is a G_l proof $\mathcal{G}'$ of S, $|\mathcal{G}'| \leq 2|\mathcal{G}| + 2\#S + 1$, $\#\mathcal{G}' \leq \#\mathcal{G} + 2\#S$, with each interior inference of $\mathcal{G}'$ an instance of Cut.*

Proof: Apply Corollary 3.11, then Corollary 3.14.

Recall that G is the system with no restrictions on its axioms or rules of inference, and is essentially equivalent to the traditional sequent calculus. Let G^- be G without the Cut rule. G^- is complete, of course.

Cook and Reckhow, in [4] and [5], investigate many different proof systems and prove polynomially bounded simulation results between pairs of systems. If $\mathcal{S}_1$ and $\mathcal{S}_2$ are two proof systems, then $\mathcal{S}_1 \leq_p \mathcal{S}_2$ means that for every proof P_2 of a formula in $\mathcal{S}_2$ there is a proof P_1 of the same formula in $\mathcal{S}_1$, with the size of P_1 bounded by a polynomial in the size of P_2. Here, "size" is interpreted to mean number of steps in a Turing-machine computation, which is essentially our #-measure. Cook and Reckhow report:

$$\text{Extended Resolution} \leq_p G \leq_p \text{Resolution, and}$$
$$\text{Extended Resolution} \leq_p G^-.$$

Here G and G^- refer to standard versions of Gentzen-style systems.

In this notation Theorems 2.10, 2.12, 3.10, and 3.13 imply:

$$\text{Resolution} \equiv_p \text{Literal Trees} \equiv_p G_l$$

where in fact the subscript p indicates a possible increase in proof length bounded by the length of the formula.

In this section we will show a one-to-one correspondence between Extended Resolution and system G, with respect to $|\cdot|$ and # measures. As with earlier results, the proofs will also provide a correspondence between the "structures" of the proofs, that is, their underlying trees, but the savings in space allowed by the extension rule precludes a corollary to the effect that Extended Resolution and G have the same computational complexity. We will return to this at the end of this section.

Just as literal trees were seen to be the "skeletons" of G_l proofs, the following notion exhibits the essential structure of a proof in system G. What we call formula trees below are broadly equivalent to a Beth/Smullyan tableaux; we modify some of the details in order to discuss more precisely their relationship to Gentzen proofs.

3.17 Definition A *formula tree* $\mathfrak{F}$ is a pair (T, f), where T is a tree and f maps nonroot nodes of T to finite sets of formulas. We insist that if a node D has children $D_1, \ldots, D_K$ for $K > 1$, then each $f(D_i)$ is a singleton.

From now on, we will use the following convention. If $f(N)$ is a singleton, we will write $f(N)$ to stand for the *formula* in the set $f(N)$. Expressions such as "$\sim f(N)$" should be taken as designating the negation of the formula which is the element of $f(N)$.

If X has children X_1 and X_2, with $f(X_1) = \sim f(X_2)$, then X is a *cut node.*

A formula tree $\mathfrak{F}$ is *closed* if, for each leaf node L, there exists $L^* \geq L$, with $f(L^*)$ containing a formula which is the negation of a formula in $f(L)$.

As usual, we define $N \simeq M$ if $Sub(N) \simeq Sub(M)$. The *weight* of a node N, $wt(N)$, is the sum of the number of formulas in the children of N. Then $|\mathfrak{F}| = \Sigma wt(N)$, the sum taken over all nodes of T, and $\#\mathfrak{F} = |\mathfrak{F}|/\simeq$.

If $\wedge\{A_1, \ldots, A_K\} \in f(N)$, we say that $\wedge\{A_1, \ldots, A_K\}$ is *analyzed* at $K < N$ if $f(K_1) = \{A_1, \ldots, A_K\}$. If $\vee\{A_1, \ldots, A_K\}$ $f(N)$, then $\vee\{A_1, \ldots, A_K\}$ is analyzed at $D < N$ if the children of D are $D_1, \ldots, D_K$, with $f(D_i) = A_i$.

We now want to define the notion of a set of formulas *occurring* on $\mathfrak{F}$, in much the same way as for literal trees, but we now must take into account the analysis of formulas. We begin by defining $Occ(N)$, for nodes N:

3.18 Definition Let $\mathfrak{F} = (T, f)$ be a formula tree

(i) if L is a leaf node, $Occ(L) = f(L)$.
(ii) $Occ(K) = [Occ(K_1) \cup \wedge f(K_1)] - f(K)$.
(iii) $Occ(D) = [\bigcup Occ(D_i)] \cup [\vee\{f(D_i)\}] - f(D)$.
(iv) $Occ(X) = Occ(X_1) \cup Occ(X_2)$.

We then let

(v) $Occ(\mathfrak{F}) = Occ(\text{root of } \mathfrak{F})$.

For example let $\mathfrak{F}$ be the tree shown in Figure 3, with $f(N)$ written below the name of the node N:

Then $Occ(L_1) = \{A\}$, $Occ(D_1) = \{(A \vee B)\}$, $Occ(D) = \{(E \vee F), (S \vee T)\}$. The formula $(A \vee B)$ has been deleted since it is $f(D)$. This reflects the fact that although $(A \vee B) \in Occ(D_1)$, the nodes below D_1 represent an analysis of the ancestor node $A \vee B$. $Occ(\mathfrak{F})$ is $\{(S \vee T), (E \vee F), (A \wedge C), ((A \vee B) \vee Y)\}$.

We can now make precise the idea that a formula tree is essentially a G-proof.

3.19 Proposition *Let $\mathcal{G}$ be a G-proof of a conjunction $\wedge\Delta$. Then there exists a closed formula tree $\mathfrak{F}$, $|\mathfrak{F}| = |\mathcal{G}|$, $\#\mathfrak{F} = \#\mathcal{G}$, with $Occ(\mathfrak{F}) \subseteq \Delta$.*

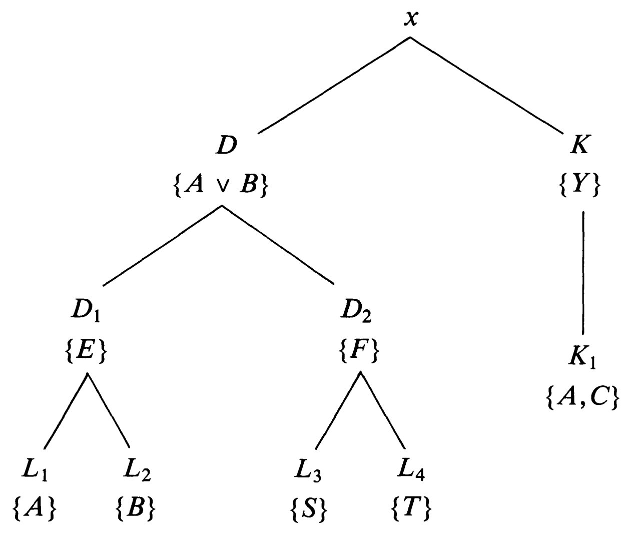


Figure 3

Proof: Suppose $\mathcal{G}$ is (T, seq). We define $\mathfrak{F}$ as (T, f), with $f(N) = \{$m.f. of $seq(N)\}$. The fact that $|\mathfrak{F}| = |\mathcal{G}|$ is immediate, and $\#\mathfrak{F} = \#\mathcal{G}$ follows from the fact that if $seq(N) = seq(M)$ in $\mathcal{G}$, then $Sub(N) = Sub(M)$ in $\mathcal{G}$, and hence $Sub(N) = Sub(M)$ in $\mathfrak{F}$.

The root of T has one child N_1, with Δ equal to the set of m.f. in $seq(N_1)$. (Actually, Δ is precisely $seq(N_1)$ in this case.) Then we can see that $\mathfrak{F}$ is closed: let L be a leaf node of T, and let A be an m.f. of $seq(L)$ such that $\sim A$ is also in $seq(L)$. (Here, we are using Lemma 3.8.) But then $\sim A$ must occur as m.f. in some $L^* > L$, since we can apply Lemma 3.7 to the G proof of Δ determined by the subtree generated by N_1 to conclude that either $\sim A$ occurs as m.f. in this tree, or is an element of Δ. But if $\sim A$ is an element of Δ, $\sim A$ is an m.f. of the node N_1, with respect to the original proof $\mathcal{G}$. In either case, then, $\sim A$ appears in $f(L^*)$ for some $L^* > L$, so the node L closes.

Finally, to see that $Occ(\mathfrak{F}) \subseteq \Delta$, suppose that some $\wedge \{A_1, \ldots, A_k\} \in Occ(\mathfrak{F})$. Then there is a node K such that $f(K_1) = \{A_1, \ldots, A_k\}$, and the formula $\{A_1, \ldots, A_K\}$ does not occur above K_1 in $\mathfrak{F}$. Then $\wedge\{A_1, \ldots, A_K\} \in seq(K)$ in $\mathcal{G}$, since this is what the m.f. $A_1, \ldots, A_K$ produce in the inference from K_1 to K. Since $\wedge\{A_1, \ldots, A_K\}$ does not appear above K in $\mathfrak{F}$, it is not an m.f. above K, hence by Lemma 3.7, $\wedge\{A_1, \ldots, A_K\} \in \Delta$. Similarly, if $\vee\{A_1, \ldots, A_K\}$ occurs at node D, then $\vee\{A_1, \ldots, A_K\} \in seq(D)$ in $\mathcal{G}$, and hence $\vee\{A_1, \ldots, A_K\} \in \Delta$.

As expected, we can construct a G-proof based on a closed-formula tree.

3.20 Proposition *If $\mathfrak{F}$ is a closed formula tree, then there is a G proof $\mathcal{G}$ of $Occ(\mathfrak{F})$ with $|\mathcal{G}| \leq |\mathfrak{F}|$, $\#\mathcal{G} \leq \#\mathfrak{F}$.*

Proof: The proof is similar to that of 3.13.

We can now parallel the course taken earlier and manipulate formula trees in order to derive results about G-proofs.

In particular, we can now show the equivalence of system G and Extended Resolution. The strategy is as follows. Given a proof $\mathcal{G}$ of Δ in system G, we consider the corresponding formula tree $\mathfrak{F}$. This is not, in general, a literal tree, so we cannot produce an ordinary resolution proof. But the thing that keeps $\mathfrak{F}$ from being a literal tree is the existence of nonliteral minor formulas at the nodes, and the extension rule is precisely the tool for replacing these by literals. We can systematically transform $\mathfrak{F}$ into a literal tree by using extension, and can then produce a resolution proof of the extended set of formulas Δ^*, that is, an extended resolution proof of Δ. The basic tree structure of the formula tree and the literal tree will be the same.

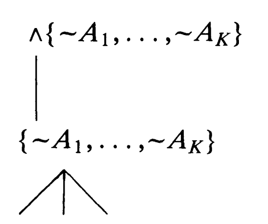
In the other direction, we can, given an Extended Resolution proof of Δ, build a G-proof of Δ with the same tree structure.

3.21 Theorem *Let $\mathfrak{F}$ be a closed formula tree, and let $Occ(\mathfrak{F}) = \Delta$. Then there is a closed literal tree $\mathfrak{I}$ for Δ^*, with Δ^* obtained from Δ by extension, $|\mathfrak{I}| = |\mathfrak{F}|$, $\#\mathfrak{I} = \#\mathfrak{F}$.*

Proof: Suppose $\mathfrak{F}$ is (T, f). If $\mathfrak{F}$ is not a literal tree, we will construct a new formula tree $\mathfrak{F}' = (T', f')$, with $Occ(\mathfrak{F}')$ obtained from $Occ(\mathfrak{F})$ by one application of extension, $|\mathfrak{F}'| = |\mathfrak{F}|$, $\#\mathfrak{F}' = \#\mathfrak{F}$, such that the number of interior nodes N' of $\mathfrak{F}'$ with $f(N')$ not a single literal is smaller than the number of such nodes in $\mathfrak{F}$.

Let N be a node of $\mathfrak{F}$ with $f(N) = \vee\{A_i\}$ or $\wedge\{\sim A_i\}$. Introduce a new literal x, replace each occurrence of $\vee\{A_i\}$ in $\mathfrak{F}$ by x, and each occurrence of $\wedge\{\sim A_i\}$ by $\bar{x}$. If $\vee\{A_i\}$ was analyzed in $\mathfrak{F}$ at some node N^-, add a new child N^-_{K+1} to N^-, with $f(N^-_{K+1}) = \bar{x}$; N_{K+1} then becomes a (closed) leaf node. If $\wedge\{\sim A_i\}$ appears on the tree, then using Lemma 3.9, we assume that $\{\sim A_1, \ldots, \sim A_K\}$ appears immediately below it in $\mathfrak{F}$.

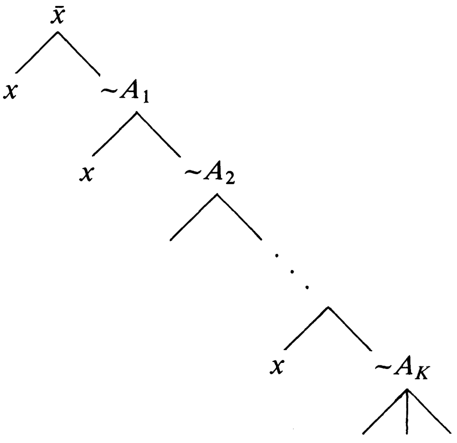
We then have the following configuration (in $\mathfrak{F}$):


$\wedge\{\sim A_1, \ldots, \sim A_K\}$
$\{\sim A_1, \ldots, \sim A_K\}$

We replace this in $\mathfrak{F}'$ as shown at the top of p. 500. This completes the definition of $\mathfrak{F}'$.

To verify that this works, first note that $|\mathfrak{F}'| = |\mathfrak{F}|$, since the new literals x, $\bar{x}$ either replace nodes in $\mathfrak{F}$ or become leaf nodes, while the new interior nodes such as the $\sim A_i$ above represent k nodes of weight 1 replacing 1 node of weight k. The construction is clearly well defined with respect to the $\simeq$ relation, so that $\#\mathfrak{F}' = \#\mathfrak{F}$. The tree $\mathfrak{F}'$ still closes since the set of formulas above any given leaf node has not changed in the passage from $\mathfrak{F}$ to $\mathfrak{F}'$.

Finally, to see that $Occ(\mathfrak{F}')$ is obtained from $Occ(\mathfrak{F})$ by extension: the for-



mulas in $Occ(\mathfrak{F}')$ not in $Occ(\mathfrak{F})$ arise from the replacement of occurrences of $\vee\{A_i\}$ by x or $\wedge\{\sim A_i\}$ by $\bar{x}$, which is clearly an instance of extension, or by the introduction of a new leaf $\bar{x}$, or the splitting of $\{\sim A_1,\ldots,\sim A_k\}$ into k new interior nodes. The adjacent nodes $A_1,\ldots,A_k$ (which, in $\mathfrak{F}$, represented the analysis of $\vee\{A_1,\ldots,A_k\}$, which has disappeared) now have a new sibling, the node labeled $\bar{x}$, thus $\vee\{\bar{x}, A_1,\ldots,A_K\} \in Occ(\mathfrak{F}')$. But this is a formula (expressing "$x \rightarrow A_1 \ldots A_K$") added to Δ in the definition of extension. In the same way, the old $\sim A_1,\ldots,\sim A_k$, the analysis of $\wedge\{\sim A_1,\ldots,\sim A_k\}$ in $\mathfrak{F}$, have been replaced by $(x \vee \sim A_1),\ldots,(x \vee \sim A_k)$ which express "$A_1 \vee \ldots \vee A_k \rightarrow x$", and are added to Δ by extension. Thus $\mathfrak{F}'$ is as claimed, and iterating the construction eventually yields a literal tree $\mathfrak{T}$.

3.22 Corollary *If $\mathfrak{G}$ is a G proof of Δ, then there exists an Extended Resolution proof $\mathfrak{R}^*$ of Δ, with $|\mathfrak{R}^*| \leq |\mathfrak{G}|$, $\#\mathfrak{R}^* \leq \#\mathfrak{G}$.*

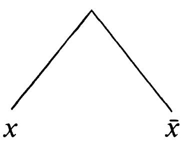
Proof: Proposition 3.19 yields a formula tree $\mathfrak{F}$ for $\mathfrak{G}$, with $Occ(\mathfrak{F}) = \Delta$. Theorem 3.21 produces a literal tree $\mathfrak{T}$, *with* $Occ(\mathfrak{T}) = \Delta^*$, Δ^* obtained from Δ by extension. Corollary 2.11 then gives a resolution proof $\mathfrak{R}$ of Δ^*, which can be considered an Extended Resolution proof $\mathfrak{R}^*$ of Δ. The $|\cdot|$ and #-measures are respected by each of the transformations.

The above procedure can be reversed, providing a method of eliminating the abbreviations in an Extended Resolution proof to produce a G-proof. Again, we perform the construction on formula trees.

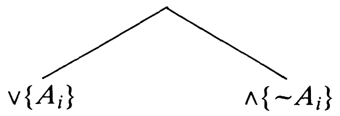
3.23 Theorem *Let $\mathfrak{T}$ be a closed cut literal tree for Δ^*, where Δ^* is a set of clauses obtained from a set of formulas Δ by extension. Then there exists a closed cut formula tree $\mathfrak{F}$ for Δ, with $|\mathfrak{F}| \leq |\mathfrak{T}|$, $\#\mathfrak{F} \leq \#\mathfrak{T}$.*

Proof: The set Δ^* is obtained from Δ by applications of extension. Thus we have a chain $\Delta = \Delta_0 \subseteq \Delta_1 \subseteq \ldots \subseteq \Delta_n = \Delta^*$, in which each Δ_{i+1} is obtained from Δ_i by one application of extension. We will show how, given a closed cut formula tree $\mathfrak{F}_{i+1}$ for Δ_{i+1}, we can build a closed cut formula tree $\mathfrak{F}_i$ for Δ_i, with $|\mathfrak{F}_i| \leq |\mathfrak{F}_{i+1}|$, $\#\mathfrak{F}_i \leq \#\mathfrak{F}_{i+1}$. This clearly suffices to prove the theorem.

Suppose Δ_{i+1} is obtained from Δ_i by substituting x for $\vee\{A_i\}$ (and thus $\bar{x}$ for $\wedge\{\sim A_i\}$). Then $\mathfrak{F}_i$ is obtained from $\mathfrak{F}_{i+1}$ by first replacing each adjacent pair



by



The only noncut nodes of $\mathfrak{F}_{i+1}$ are leaf nodes. The only occurrences of x or $\bar{x}$ among the leaf nodes are instances of replacing $\vee\{A_i\}$ by x, replacing $\wedge\{\sim A_i\}$ by $\bar{x}$, or occurrences of the new clauses $\bar{x} \vee A_1 \vee \ldots \vee A_K$, $x \vee \sim A_1, \ldots, x \vee \sim A_k$. This is because x has no occurrences in Δ_i, and the above are the only type of formula in $\Delta_{i+1} - \Delta_i$. Any occurrence of $x[\bar{x}]$ which arose due to a replacement of $\vee\{A_i\}[\wedge\{\sim A_i\}]$ maintains this relationship of a formula and its negation.

The other occurrences of x or $\bar{x}$ are due to the introduction of the formulas $\bar{x} \vee A_1 \vee \ldots \vee A_K$ and $x \vee \sim A_1, \ldots, x \vee \sim A_k$. Whenever $\bar{x} \vee A_1 \vee \ldots \vee A_K$ appears, simply delete the $\bar{x}$ node. This changes the formula occurring at the parent of this leaf to $A_1 \vee \ldots \vee A_k$, which is not necessarily a formula in Δ_i, but, since the node $\bar{x}$ was a leaf, hence closed in $\mathfrak{F}_{i+1}$, there was a node N above $\bar{x}$ with $f(N) = x$. But then in $\mathfrak{F}_i$, $f(N)$ is $\vee\{A_i\}$, so that the nodes labeled A_i represent an analysis of $\vee\{A_i\}$, and do not contribute to $Occ(\mathfrak{F}_i)$.

Finally, if x occurs at a leaf adjacent to one of the $\sim A_i$ (corresponding to a new clause $x \vee \sim A_i$ in Δ_{i+1}), delete the nodes with x and change the $\sim A_i$ node to contain all of the $\sim A_1, \ldots, \sim A_K$. These new elements do not affect the closing of the tree, and since the x node was closed in $\mathfrak{F}_{i+1}$, there was an $\bar{x}$ above it in $\mathfrak{F}_{i+1}$, thus an $\wedge\{\sim A_i\}$ in $\mathfrak{F}_i$, and $\{\sim A_1, \ldots, \sim A_K\}$ is the analysis of this formula.

No new nodes have been added anywhere, thus $|\mathfrak{F}_{i+1}| \leq |\mathfrak{F}_i|$ and $\#\mathfrak{F}_{i+1} \leq$

3.24 Corollary *Given any Extended Resolution proof* $\mathcal{R}^*$ *of a set* Δ, *there is a G-proof* $\mathcal{G}$ *of* Δ, *with* $|\mathcal{G}| \leq |\mathcal{R}|$, $\#\mathcal{G} \leq \#\mathcal{R}$.

Proof: Immediate from Theorem 3.23 and Proposition 3.20.

When the extension rule is iterated, very long formulas can be abbreviated as single literals. In this way an Extended Resolution proof could conceivably be a much smaller "object" than the corresponding G proof. Corollary 3.24 shows that the underlying trees and the relationships among the clauses/sequents in the proof will be almost identical. However, the space required to represent the sequents in their "unabbreviated" state might be large compared to the size of the Extended Resolution proof. (The question is open.)

Thus, Theorem 3.23 and Corollary 3.24 cannot be construed as providing efficient simulations of Extended Resolution by G proofs, when these are considered as nondeterministic algorithms. This is the only place in the paper where the results do not translate immediately to the complexity-theoretic setting.

We can conclude, though, that there is no polynomial relating the number of steps in (shortest) G-proofs to the number of steps in G_l-proofs:

3.25 Corollary *For the sets of clauses S_n of Theorem* 2.14 (*considered as sequents*), *and for some $c > 0$, the shortest G_l proof of S_n has at least 2^{cn} inferences, while there exist G-proofs of S_n with $O(n^4)$ inferences.*

Proof: From 2.14, 2.16, 3.11, and 3.24.

3.26 Corollary *Given any G-proof $\mathcal{G}$ of Δ, there is a proof $\mathcal{G}'$ of Δ, with $|\mathcal{G}'| \leq 2|\mathcal{G}| + 1$, $\#\mathcal{G}' \leq 2\#\mathcal{G}$, such that every inference at interior nodes of $\mathcal{G}'$ is an instance of cut.*

Proof: From $\mathcal{G}$, we get an Extended Resolution proof $\mathcal{R}^*$ of Δ, by Corollary 3.22, which is in fact a Resolution proof of Δ^*. Theorem 2.12 provides a literal tree $\mathfrak{J}$ for Δ^*, with $|\mathfrak{J}| \leq 2|\mathcal{G}| + 1$, $\#\mathfrak{J} \leq \#\mathcal{G} + \#\Delta$. Note that $\#\Delta \leq \#\mathcal{G}$, then apply Theorem 3.23 and Proposition 3.20.

It is well-known that proofs in system G can be transformed into cut-free proofs. The usual cut-elimination procedure involves a potential growth in the size of the proof as cuts are eliminated. Statman [13], investigating the tree size of Gentzen proofs (our $|\cdot|$ measure), showed that this cannot always be avoided, by exhibiting a class of formulas whose shortest cut-free proofs had exponential length but which had polynomial-bounded proofs when cut was allowed. In other words, the system G without cut cannot polynomially simulate G.

Corollaries 3.16 and 3.26 assert that eliminating cuts can *never* substantially shorten proofs, that the shortest proofs of a formula (up to a factor of 2) are "pure cut" proofs.

REFERENCES

[1] Beth, E. W., "Semantic entailment and formal derivability," *Mededelingen Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde, Nieuwe Serie*, N. S. 19 (1955), pp. 309–342.

[2] Cook, S. A., "Feasibly constructive proofs and the propositional calculus," *Proceedings of the Seventh Annual ACM Symposium on Theory of Computing*, Association for Computing Machinery, 1975, pp. 83–97.

[3] Chang, C. L. and R. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.

[4] Cook, S. A. and R. A. Reckhow, "On the lengths of proofs in the propositional calculus," *Proceedings of the Sixth Annual ACM Symposium on Theory of Com-*

puting , Association for Computing Machinery (1974), pp. 135–148. See also corrections for the above in *SIGACT News*, vol. 6, no. 3 (1974), pp. 15–22.

[5] Cook, S. A. and R. A. Reckhow, "The relative efficiency of propositional proof systems, *The Journal of Symbolic Logic*, vol. 44 (1979), pp. 36–50.

[6] Galil, Z., "On the complexity of regular resolution and the Davis-Putnam procedure," *Theoretical Computer Science*, vol. 4 (1977), pp. 23–46.

[7] Galil, Z., "On resolution with clauses of bounded size," *SIAM Journal on Computing*, vol. 6 (1977), pp. 444–459.

[8] Haken, A., "The intractability of resolution," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 1984.

[9] Hintikka, J., "Form and content in quantification theory," *Acta Philosophica Fennica*, vol. 8 (1955), pp. 11–55.

[10] Kleene, S. C., *Introduction to Mathematical Logic*, Van Nostrand, 1967.

[11] Kowalski, R., and P. Hayes, "Semantic trees in automatic theorem proving," pp. 87–101 in *Machine Intelligence*, vol. 4, eds. B. Meltzer and D. Michie, American Elsevier Publishing Company, New York, 1969.

[12] Robinson, J. A., "A machine oriented logic based on the resolution principle," *Journal of the Association for Computing Machinery*, vol. 12 (1965), pp.23–41.

[13] Statman, R., "Bounds for proof-search and speed-up in the predicate calculus," *Annals of Mathematical Logic*, vol. 15 (1978), pp. 225–287.

[14] Schutte, K., "Ein System des verknupfenden Schliessens," *Archiv für Mathematische Logik und Grundlagenforsch*, vol. 2 (1956), pp. 55–67.

[15] Smullyan, R. M., *First Order Logic*, Springer-Verlag, Berlin, 1968.

[16] Tseitin, G. S., "On the complexity of derivations in propositional calculus," in *Studies in Constructive Mathematics and Mathematical Logic, Part II*, ed. A. O. Slisenko, 1968.

Department of Mathematics
Wesleyan University
Middleton, CT 06457