The Complexity of Analytic Tableaux

Noriko H. Arai[*]
Department of Computer Science
Hiroshima City University
narai@cs.hiroshima-cu.ac.jp

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

Alasdair Urquhart[‡]
Department of Philosophy
University of Toronto
urquhart@cs.toronto.edu

ABSTRACT

The method of analytic tableaux is employed in many introductory texts and has also been used quite extensively as a basis for automated theorem proving. In this paper, we discuss the complexity of the system as a method for refuting contradictory sets of clauses, and resolve several open questions. We discuss the three forms of analytic tableaux: clausal tableaux, generalized clausal tableaux, and binary tableaux. We resolve the relative complexity of these three forms of tableaux proofs and also resolve the relative complexity of analytic tableaux versus resolution. We show that there is a quasi-polynomial simulation of tree resolution by analytic tableaux; this simulation cannot be improved, since we give a matching lower bound that is tight to within a polynomial.

1. INTRODUCTION

The method of analytic tableaux, or truth trees, is employed in many introductory logic texts; it is given a particularly elegant formulation in Smullyan's monograph [9]. It has also been used quite extensively as a basis for automated theorem proving. In this paper, we discuss the complexity of the system as a method of refuting contradictory sets of clauses. One of the results of the analysis is that the complexity depends crucially on the way in which the clauses are represented, and on the exact form of the decomposition rules employed.

An analytic tableaux tree for a CNF formula f has a root node labelled by the initial clauses of f. For each internal node k of the tree, a particular clause labelling k is chosen and decomposed into l disjoint subclauses; each of the l children of k are labelled with all of the clauses labelling k plus one of the l disjoint subclauses. A path in the proof tree is closed if the leaf node is labelled with a literal and its negation (and other clauses as well). An analytic tableaux tree for f is a refutation of f if all paths are closed.

There are various possible forms of the tableau decomposition rule, leading to different versions of analytic tableaux. The simplest form of the rule, leading to what we call *clausal tableaux* below, is one in which the decomposition of a clause into all of its component literals must be carried out in a single step. A general form of the rule, leading to what we call *generalized clausal tableaux*, allows the decomposition of a clause AB into two disjoint clauses A and B. As we shall see, the more general form of the clausal rule can result in drastically reduced complexity. One can also consider clauses as formulas built up from literals, using disjunction as a binary connective. The tableau rule then takes the form where a formula $A \vee B$ is decomposed into its immediate subformulas A and B. This form of analytic tableaux, the one used in Smullyan's monograph [9], we call *binary tableaux.*

The relative complexity of analytic tableaux versus tree resolution is interesting since both of these systems are used extensively in automated theorem proving. It has been known since the work of Cook and Reckhow [3] that there are sets of clauses that are easy to refute for tree resolution but hard for analytic tableaux. As a result, the question of the relative complexity of these two systems was generally held to be settled for many years. However, the question was reopened by Massacci [7, 6] who presented efficient binary analytic tableaux proofs for Cook and Reckhow's examples, thus pointing out that the lower bounds of [3] only hold for clausal analytic tableaux. Our paper gives a conclusive answer to this question of relative complexity and also shows the startling sensitivity of the analytic tableaux method to the form of the tableaux decomposition rule.

The paper is organized as follows. In §2, we present the lower bound of Cook and Reckhow for clausal tableaux, with improved parameters. In §3, we show that the speed-up observed by Massacci for binary analytic tableaux (for these examples) is highly dependent on the order in which the clauses are parenthesized. In particular, with one order of bracketing, we can construct quasi-polynomial size refutations, while with another order, exponential-size refutations are required.

This exponential separation does not hold if we adopt the generalized clausal form of the decomposition rule. In §4, we show that generalized clausal tableaux can quasi-polynomially simulate tree resolution. This generalizes the

[*]Research supported by the Ministry of Education of Japan.
[†]Research supported by NSF grant CCR-9820831, US-Israel BSF grant 98-00349, and an NSERC grant.
[‡]Research supported by NSERC.

STOC'01, July 6-8, 2001, Hersonissos, Crete, Greece.

observation of Massacci about the speed-up of binary tableaux over clausal tableaux. Finally in §5, we show that the simulation in §4 is essentially tight by giving a family of formulas with linear size tree-resolution proofs, but requiring quasi-polynomial size generalized clausal tableaux proofs.

Combining these results we obtain a nearly complete picture of the relative complexity of the various forms of analytic tableaux and resolution: Tree resolution polynomially simulates generalized clausal analytic tableaux, but the reverse simulation does not hold; generalized clausal analytic tableaux polynomially simulates both binary and clausal analytic tableaux, but the reverse simulations do not hold.

If Σ is a set of clauses and l is a literal, then we write $\Sigma\lceil l$ for the set of clauses that results from Σ by removing all clauses containing l and deleting $\overline{l}$ from any clause in which it occurs. We will use trees in describing both derivations in proof systems and in defining sets of clauses. If T is a binary tree with immediate subtrees U and V, then we write $T = U \oplus V$.

2. CLAUSAL TABLEAUX

In this section, we consider the simple form of the tableau where all formulas are clauses, and the tableau rule requires a clause to be decomposed into its component literals in one step. If Σ is a set of clauses, then a *clausal tableau for* Σ is a tree in which the interior nodes are associated with clauses from Σ; if a node is associated with a given clause, then the children of that node are labelled with the literals in the clause (the root of the tree remains unlabelled). (In the remainder of this section, the word "tableau" should be understood as referring to a a clausal tableau.) A tableau for Σ is a *refutation of* Σ if every branch in the tableau is closed (i.e. contains a literal and its negation). We shall also count a single unlabelled node as a refutation of the set of clauses $\{\Lambda\}$.

We define the *size* of a tableau refutation as the number of interior nodes in the tableau (this measure of complexity, omitting the leaves of the tree, is convenient for inductive proofs). If T is a tree in which every interior node has at most k children, then the number of leaves of T is bounded by $I(T)\cdot(k-1)+1$, where $I(T)$ is the number of interior nodes in T; if every interior node of T has exactly k children, then the number of leaves of T is exactly $I(T)\cdot(k-1)+1$. If Σ is a set of clauses, then $t(\Sigma)$ is defined to be the minimum size of a tableau refutation of Σ. Because of the simple structure of clausal tableau refutations, it is possible to prove exact lower bounds on their complexity. The basic tools are the following lemmas.

LEMMA 2.1. *In a tableau refutation of minimal size, no branch contains repeated literals.*

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

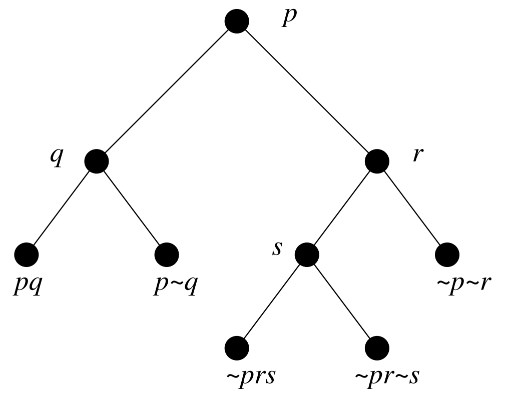

Figure 1: $\Sigma(T) = \{pq, p{\sim}q, {\sim}prs, {\sim}pr{\sim}s, {\sim}p{\sim}r\}$

LEMMA 2.2. *If Σ is an unsatisfiable set of clauses, then $t(\Sigma)$ satisfies the recursive equation*

$$t(\Sigma) = \min\{t(\Sigma\lceil l_1) + \cdots + t(\Sigma\lceil l_n) + 1 : l_1 \vee \cdots \vee l_n \in \Sigma\}.$$

Proof. For $C = l_1 \vee \cdots \vee l_n \in \Sigma$, let T be a tableau refutation of Σ that is minimal among refutations that have C associated with their root. Let $T_1, \ldots, T_n$ be the immediate subtrees of T having $l_1, \ldots, l_n$ as labels on their roots. By Lemma 2.1, the literal l_i does not occur in T_i below the root of T_i; the complement of l_i may occur as the label of at least one leaf in T_i. Thus if we remove from T_i the leaves labelled with $\overline{l_i}$, the result is a refutation of $\Sigma\lceil l_i$. Hence the size of T_i is $t(\Sigma\lceil l_i)$, so that the size of T is

$$t(\Sigma\lceil l_1) + \cdots + t(\Sigma\lceil l_n) + 1.$$

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

Although clausal tableaux work well on simple examples, there are cases where any tableau refutation necessarily contains a great deal of repetition. This is shown by a set of examples due to Cook [1]. Cook's construction associates a set of clauses with a labelled binary tree as follows. Let T be a binary tree in which the interior nodes are labelled with distinct variables. We associate a set of clauses $\Sigma(T)$ with T, in such a way that each branch b in $\Sigma(T)$ has a clause $C_b \in \Sigma(T)$ associated with it. The variables in C_b are those labeling the nodes in b; if P is such a variable, then P is included in C_b if b branches to the left below the node labelled with P, otherwise C_b contains ${\sim}P$. Figure 2 shows a simple example.

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

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

The next lemma allows us to compute $t(T) = t(\Sigma(T))$ directly from the structure of T. It shows that for the set of clauses $\Sigma(T)$, there is an optimal refutation that is essentially unique.

LEMMA 2.3. *The function $t(T)$ satisfies the following recursion equations:*

1. *If T has only one node, then $t(T) = 1$;*
2. *If $T = U \oplus V$, then $t(T) = t(U)t(V) + \min\{t(U), t(V)\}$.*

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

Assume the recursion equations hold for trees of size less than that of T, and let $T = U \oplus V$. Let $C = l_1 \vee \cdots \vee l_k$ be a clause in $\Sigma(T)$ that is associated with a branch ending in a leaf in U (the argument for branches in V is symmetrical). Define U_j for $2 \leq j \leq k$ to be the labelled tree $U \restriction l_j$. Let $t_C(T)$ be the size of a minimal tableau in which C is associated with the root. Then by Lemma 2.2,

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

By Lemma 2.2 again,

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

so by (1),

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

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

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

completing the proof. □

THEOREM 2.4. *For $n > 0$,*

1. *The clauses Σ_n satisfy the recursion equations: $t(\Sigma_0) = 1$, $t(\Sigma_{n+1}) = t(\Sigma_n)[t(\Sigma_n) + 1]$;*
2. *$t(\Sigma_n) = \lfloor 2^{c2^n} \rfloor$, where $0.67618 < c < 0.67819$.*

Proof. The left and right subtrees of the complete binary tree T_n are isomorphic, so the first claim follows immediately from Lemma 2.3.

Let $z_n = t(\Sigma_n)$; we can estimate the size of z_n by using Aho and Sloane's analysis of doubly exponential sequences [4, §2.2.3]. Taking logarithms to the base 2, we have by the first part of the lemma,

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

hence unrolling the recursion,

$$\log z_n = 2^n \sum_{k=0}^{n-1} \alpha_k 2^{k+1},$$

where $\alpha_k = \log(1 + 1/z_k)$. Defining

$$c_n = \sum_{k=0}^{n-1} \alpha_k 2^{k+1}, \quad c = \sum_{k=0}^{\infty} \alpha_k 2^{k+1},$$

we set $r_n = 2^n(c - c_n)$. We have

$$r_n = \sum_{k=0}^{\infty} \alpha_{n+k} 2^{k+1} < \alpha_n,$$

so $2^{r_n} < 2^{\alpha_n} = 1 + 1/z_n$. Hence, $2^{c2^n} - z_n = z_n(2^{r_n} - 1) < 1$, showing that $z_n = \lfloor 2^{c2^n} \rfloor$. The series for c converges rapidly; we find $c = 0.67618634966\ldots$ from $n = 5$. □

An exponential lower bound for the sets of clauses Σ_n was first stated (without proof) in [2]; the proof above is based on joint work of Cook and Urquhart, and is an improved version of the proof published in Urquhart [10]. The first published proof of the lower bound for clausal tableaux was given by Murray and Rosenthal [8], who gave a lower bound of $2^{2^{n-1}}$ for the size of clausal tableau refutations of Σ_n.

Theorem 2.4 has some significance for automated theorem proving based on clausal tableau methods. The set Σ_6 contains only 64 clauses of length 6, but the minimal tableau refutation for Σ_6 has 10,650,056,950,806 interior nodes. This shows that any practical implementation of the clausal tableau method should incorporate routines to eliminate repetition in tableau construction. This applies in particular to the model-elimination method [5], which can be considered as a version of clausal tableaux.

3. BINARY TABLEAUX

In recent work [7, 6] Massacci has shown that the sets of clauses Σ_n discussed in the previous section in fact have quasi-polynomial size analytic tableau refutations, provided we use binary tableaux. In this formulation, formulas are built up from binary disjunctions, and in an application of the tableau rule, formulas are decomposed into their immediate subformulas, rather being decomposed into their component literals, as is required in the case of clausal tableaux. Massacci's result revealed an unsuspected gap between these two formulations of the tableau system, and reopened the question of the relative complexity of the tableau system and resolution (in the form where resolution refutations are represented as trees).

A careful examination of Massacci's proofs reveals that they are strongly dependent on the way in which the formulas are parenthesized. In the case of the sets of clauses Σ_n, it is obvious that some variables are more significant than others. The variable labelling the root of the tree T_n occurs in all of the clauses in Σ_n, whereas a variable labelling a node immediately above a leaf occurs in only two clauses. Hence, assigning a value to the top variable results in a drastic reduction in the complexity of the problem since the set of clauses is cut in half. However, setting a bottom variable results in only a tiny amount of progress. It is precisely for this reason that these examples are hard for clausal tableaux, since the decomposition rule forces us to assign values to variables low down in the tree T_n.

However, Massacci parenthesizes the formulas in such a way that all the clauses are given in the form $l_1 \vee (l_2 \vee \ldots l_n)\ldots)$, where the sequence of variables follows the sequence of nodes from the root of T_n to its leaves. This means that in a binary tableau refutation of Σ_n, we can give preference to the more significant variables. In this way, Massacci is able to construct binary tableau refutations of Σ_n of size bounded by $2^{O(n^2)}$, that is to say, a refutation of size quasi-polynomial in the size of the set of input clauses.

The observations above on the relative significance of variables suggest that we could prove a lower bound for binary tableaux similar to the lower bound of the preceding section if we parenthesized the set Σ_n in the opposite way, that is to say, if we represented the clauses in the form $(\ldots(l_1 \vee l_2) \vee \ldots l_{n-1}) \vee l_n$ This order of variables forces us to deal with the least significant variables first, and so should require a tableaux of a size comparable to that necessary in the case of clausal tableaux. In this section, we verify this conjecture.

By a *formula*, we mean a disjunction built up from literals using a binary disjunction connective. If Π is a set of formulas, then a *binary tableau for* Π is a tree in which the interior nodes are associated with formulas, in accordance with the following rules. If a formula A is associated with an internal node d, then it is either an initial formula belonging to Π, or occurs as a label on the branch determined by d; we say that A is the formula *decomposed at the node* d. If a node is associated with a given formula of the form $A \vee B$, then the children of that node are labeled with A and B respectively; we define the node labelled with A to be the *elder child*, and the node labelled with B to be the *younger child*. If an interior node d has a literal l associated with it, then the unique child of that node is labelled with l and defined to be the younger child of d. As in the case of clausal tableaux, the root of the tree is unlabelled. (In the remainder of this section, the phrase word "tableau" should be understood as referring to a binary tableau.) A tableau for Π is a *refutation of* Π if every branch in the tableau is closed (i.e. contains a literal and its negation).

Because of the more intricate nature of the binary decomposition rule, it is a little harder to analyse the complexity of binary tableaux than that of clausal tableaux. To make the recursion step in the lower bound proof easier, we define a complexity measure for binary tableaux that counts some, but not all, of the interior nodes. Let us define a node in a binary tableaux to be a *significant node* if it is either the root or a younger child of a node in the tableau. We define the *size* of a binary tableau refutation as the number of significant interior nodes in the tableau (as in the previous section, we define the unique node of a one-node tree to be an interior node). If Π is a set of clauses, then $u(\Pi)$ is defined to be the minimum size of a tableau refutation of Π.

Before proceeding to the lower bound, we require two lemmas on binary tableaux that are analogues of Lemmas 2.1 and 2.2. If A and B are formulas, then we say that A *subsumes* B if A is a subformula of B; for example, the formula $(p \vee {\sim}q)$ subsumes $(r \vee s) \vee (p \vee {\sim}q)$.

LEMMA 3.1. *If $\mathcal{T}$ is a binary tableau refutation of a set of formulas $\Pi \cup \{A, B\}$, where A subsumes B, then there is a tableau refutation $\mathcal{T}'$ of $\Pi \cup \{A\}$, where the size of $\mathcal{T}'$ is no bigger than the size of $\mathcal{T}$.*

Proof. Let Π be any contradictory set of formulas having a formula A as a member, and let C be a formula subsumed by A. We argue by induction on the complexity of C that if $\mathcal{T}$ is any tableau refutation of Π, and C occurs as a label on a node in $\mathcal{T}$, then we can shorten $\mathcal{T}$ by removing all nodes labelled with C.

Suppose that C is A itself, and that A labels a node e with a parent node d. Remove the subtree having the sibling of e as the root (if any), and then merge the nodes d and e. Remove the label A, retaining the original label on d (if there was one) as the label on the merged node. The result is still a closed tableau; the label A was never needed to justify a decomposition in $\mathcal{T}$, since it was a member of Π.

Assume that the Lemma holds for formulas subsumed by A, up to a given complexity, let B be subsumed by A, and suppose that $B \vee C$ labels a node e in the refutation with parent node d. By inductive assumption, there is no node in $\mathcal{T}$ labelled with B, from which it follows that in the subtree with e as root, $B \vee C$ is never decomposed. Remove the subtree having the sibling of e as the root (if any), and then merge the nodes d and e. Remove the label $B \vee C$, retaining the original label on d (if there was one) as the label on the merged node. The result is still a closed tableau, since the label $B \vee C$ is never needed to justify a decomposition in the subtree with e as root. $\square$

We now define the sequence $\{\Pi_n\}$ of sets of formulas used in proving the lower bound for binary tableaux. It is essentially the same as the sequence $\{\Sigma_n\}$ used in the previous section, except that we parenthesize the clauses in a particular way. More generally, we associate a set of formulas $\Pi(T)$ with any binary tree T whose internal nodes are labelled with distinct variables. Let $l_1 l_2 \ldots l_k$ be a clause in $\Sigma(T)$, where the order of literals follows the order of the edges in the corresponding branch in T from the root to the leaf. Then the corresponding formula in $\Pi(T)$ is $(\ldots(l_1 \vee l_2)\ldots) \vee l_k$. We define Π_n to be the set of formulas $\Pi(T_n)$, where T_n is the complete binary tree of depth n. Since the size of a minimal tableau refutation of $\Pi(T)$ depends only on the structure of T, we shall write $u(T)$ as an abbreviation for $u(\Pi(T))$.

To prove a lower bound for Π_n, we first require a lemma that corresponds to Lemma 2.2. If Π is a set of formulas, and C is a disjunction that is a subformula of a formula in Π, then we define $\Pi[C]$ to be the set of formulas resulting from Π by removing all formulas subsumed by C, and replacing them by C. If Π is a contradictory set of formulas, and $C \in \Pi$, then we define $u(\Pi, C)$ to be the minimal size of a binary tableau refutation of Π, in which the formula decomposed at the root of the tableau is C. Obviously, we have $u(\Pi) = \min\{u(\Pi, C) : C \in \Pi\}$.

If T is a labelled binary tree, and e a leaf in T, then we write $u(T, e)$ for $u(\Pi(T), E)$, where E is the formula in $\Pi(T)$

associated with the branch in T ending in e. For a labelled tree T, we have $u(T) = \min\{u(T,e) : e \text{ a leaf in } T\}$.

LEMMA 3.2. *If Π is a contradictory set of formulas, and $A \vee l$ is a formula in Π, where l is a literal, then $u(\Pi, A \vee l) = u(\Pi[A]) + u(\Pi \restriction l)$, if A is not a literal, while $u(\Pi, A \vee l) = u(\Pi \restriction A) + u(\Pi \restriction l)$ if A is a single literal.*

Proof. Suppose we are given a tableau refutation of Π of minimal size among tableaux in which $A \vee l$ is decomposed at the root. The size of the sub-tableau with l labelling its root is $u(\Pi \restriction l)$; this follows by the same argument as that used in Lemma 2.2.

If A is not a single literal, then the lefthand sub-tableau with A labelling its root is a refutation of $\Pi \cup \{A\}$, after the deletion of the label on its root (since the label on the root cannot be used to close a branch). By Lemma 3.1, a minimal tableau refutation of $\Pi \cup \{A\}$ is a refutation of $\Pi[A]$. The equation of the lemma follows because the node labelled with A does not contribute to the size of the refutation. On the other hand, if A is a single literal, then the size of the tableau is $u(\Pi \restriction A) + u(\Pi \restriction l)$, again by the argument of Lemma 2.2. □

As in the case of clausal tableaux, we can gain insight into the structure of tableau refutations of sets of formulas $\Pi(T)$ associated with labelled binary trees by describing logical operations on such sets of formulas in terms of operations on the underlying trees. Let T be a binary tree in which the interior nodes are labelled with distinct variables, and let e be a leaf in T. Let f be the sibling of e, and d their common parent. Then T_e^1 is the labelled tree obtained from T by removing the nodes e and f, and deleting the label on d, while T_e^2 is the tree obtained from T by removing e, removing the label on f, and then merging the nodes d and f (the label on the merged node is the same as the original label on d). Let $A \vee l$ be the formula associated with the branch in T that ends in the leaf e. Then it is easy to check that $\Pi(T)[A] = \Pi(T_e^1)$ and that $\Pi(T) \restriction l = \Pi(T_e^2)$.

LEMMA 3.3. *The function $u(T)$ satisfies the following recursion equations:*

1. *If T has only one node, then $u(T) = 1$;*
2. *If $T = V \oplus W$, and one of them, say V, is a one-element tree, then $u(T) = u(W) + 1$;*
3. *If $T = V \oplus W$, and neither V nor W is a one-element tree, then $u(T) = u(V)u(W)$.*

Proof. If T has only one node, then $\Pi(T) = \{\Lambda\}$, so $u(T) = 1$, since by our conventions, the unique node in a one-element tree counts as a significant node.

In the second case, let p be the variable labelling the root of T, and suppose that U is the lefthand subtree of T. In any refutation of $\Pi(T)$, there must be at least one node in the refutation labelled with p, from which it follows that $u(T) > u(W)$. Since there is clearly a refutation of size $u(W) + 1$, this must be a refutation of minimal size.

In the third case, let $\mathcal{T}$ be a refutation of $\Pi(T)$ in which the formula decomposed at the root of the tableau is associated with the branch ending in a leaf e. We assume that e belongs to the subtree V (the case for W is symmetrical). By Lemma 3.2, and the induction hypothesis, we have

$$\begin{aligned} u(T,e) &= u(T_e^1) + u(T_e^2) \\ &= u(V_e^1 \oplus W) + u(V_e^2 \oplus W) \\ &= u(V_e^1)u(W) + u(V_e^2)u(W) \\ &= (u(V_e^1) + u(V_e^2))u(W) \\ &= u(V,e)u(W). \end{aligned}$$

(We have assumed aboved that neither V_e^1 nor V_e^2 are one-element trees, but the reader can verify that the equation $u(T,e) = u(V,e)u(W)$ also holds when one or both are one-element trees.) We can minimize $u(V,e)u(W)$ by minimizing $u(V,e)$ over all leaves in T. Hence, $u(T) = u(V)u(W)$. □

THEOREM 3.4. *For $n > 0$,*

1. *The clauses Π_n satisfy the recursion equations $u(\Pi_1) = 2, \quad u(\Pi_{n+1}) = u(\Pi_n)^2$;*
2. *Any binary tableau refutation of Π_n must contain at least $2^{2^{n-1}}$ nodes.*

Proof. The theorem follows immediately from Lemma 3.3. □

COROLLARY 3.5. *Binary analytic tableaux cannot p-simulate tree resolution.*

Proof. The sets of clauses Σ_n have linear-size tree resolution refutations; in fact, we can label the nodes of the labelled tree T with clauses so that the result is a tree refutation of the corresponding set of clauses $\Pi(T)$. □

4. GENERALIZED CLAUSAL TABLEAUX

The complexity gap between clausal and binary tableaux shows that it makes a very large difference in the complexity of proofs if we can give preference to certain variables; with the right choice of variables, we can produce an exponential speed-up over the simple clausal rule. This suggests a still more general form of tableau rule. In this formulation, we treat disjunctions of literals as clauses (that is, sets of literals), but instead of insisting on decomposing a clause in one step, we allow arbitrary decompositions of clauses into two subclauses. This form of the rule is clearly more powerful and flexible than either the simple clausal rule, or the binary rule. We shall use the phrase "generalized clausal tableaux" to describe tableaux constructed according to this rule.

If Σ is a set of clauses, then a *generalized tableau for* Σ is a tree in which an interior node d has associated with it a clause that is either a clause from Σ or occurs as a label on the branch ending in d. If a node is associated with a given clause C, then the children of the node are labelled with clauses C_1 and C_2, where C_1 and C_2 form a partition of C (the root of the tree remains unlabelled). If C consists of a single literal, then the unique child of the node associated with C is labelled with C itself. A tableau for Σ is a *refutation of* Σ if every branch in the tableau is closed (i.e. contains a literal and its negation).

We define a complexity measure for generalized clausal tableaux that is essentially the same as the measure used for clausal tableaux. That is to say, we define the complexity of a generalized clausal tableau to be the number of interior nodes in the tableau. If Σ is a set of clauses, then $g(\Sigma)$ is defined to be the minimum size of a generalized tableau refutation of Σ.

A *tree resolution refutation* of a contradictory set of clauses Σ is a binary tree in which the leaves are labelled with clauses from Σ, the root is labelled with the empty clause Λ, and an interior node in the tree is labelled with a resolvent of the two clauses labelling its children. We define the complexity of a tree resolution refutation to be the number of leaves in the underlying tree. The *tree resolution complexity* of Σ, $r(\Sigma)$, is the minimal complexity of a tree resolution refutation of Σ.

We now prove a theorem that shows that the quasi-polynomial simulation of tree resolution by binary resolution shown by Massacci in the case of the examples of Cook and Reckhow can be generalized to a uniform quasi-polynomial simulation of tree resolution by general clausal tableaux. In the following theorem, all logarithms are to the base 2.

THEOREM 4.1. *If Σ is a contradictory set of clauses, then*

$$g(\Sigma) \leq r(\Sigma) \cdot |\Sigma|^{\log r(\Sigma)} = 2^{(\log r(\Sigma))(\log |\Sigma|+1)}.$$

Proof. The upper bound is proved by the recursive construction of a tableau refutation of Σ, starting from a tree resolution refutation of Σ. The proof is by induction on the number of variables in Σ. If Σ contains no variables, then $\Sigma = \{\Lambda\}$, $r(\Sigma) = g(\Sigma) = 1$, and the inequality is satisfied.

Assume that the upper bound holds for all sets of clauses with at most n variables, and let Σ contain $n+1$ variables. Let T be a minimal-size tree resolution refutation of Σ, with immediate subtrees U and V. Then there is a literal l so that U is a resolution proof of $\bar{l}$ of complexity $r(\Sigma \upharpoonright l)$, while V is a resolution proof of l of complexity $r(\Sigma \upharpoonright \bar{l})$; for a proof of this fact, see [10, §5]. We assume that the complexity of U is less than or equal to the complexity of V, so that $\log r(\Sigma \upharpoonright l) \leq \log r(\Sigma) - 1$.

Two cases arise in the inductive step, the first where U is a one-element tree, and the second where it has more than one node. In the first case, $\Sigma \upharpoonright l$ consists of the single literal $\bar{l}$. In this case, we construct the tableau refutation by starting from a minimal-size tableau refutation of $\Sigma \upharpoonright \bar{l}$, adding $\bar{l}$ as a label on the root, then adding a new unlabelled root, together with appropriate leaves labelled with the literal l. The result is a refutation of Σ, and it has complexity

$$\begin{aligned} g(\Sigma \upharpoonright \bar{l}) + 1 &\leq r(\Sigma \upharpoonright \bar{l}) \cdot |\Sigma \upharpoonright \bar{l}|^{\log r(\Sigma \upharpoonright \bar{l})} + 1 \\ &\leq (r(\Sigma \upharpoonright \bar{l}) + 1) \cdot |\Sigma|^{\log r(\Sigma)} \\ &= r(\Sigma) \cdot |\Sigma|^{\log r(\Sigma)} \end{aligned}$$

We now consider the second case, where the lefthand subtree U has at least two leaves. Let Σ_l be the set of all clauses in Σ containing the literal l; let $D_1 l, \ldots, D_k l$ be a list of all clauses in Σ_l. Since Σ is contradictory, $k < |\Sigma|$. Construct a generalized analytic tableau by decomposing $D_1 l$ at the root into D_1 and l; continue the construction by decomposing $D_2 l$ in the same way at the node labelled with D_1, $D_3 l$ at the node labelled D_2, and so on. Complete the construction by appending a tableau refutation of size $g(\Sigma \upharpoonright l)$ to all of the nodes labelled l, and a tableau of size $g(\Sigma \upharpoonright \bar{l})$ to the node labelled D_k. (We can construct appropriate refutations of this size by adding some leaves to tableau refutations of $\Sigma \upharpoonright l$ and $\Sigma \upharpoonright \bar{l}$ respectively.) See Figure 2.

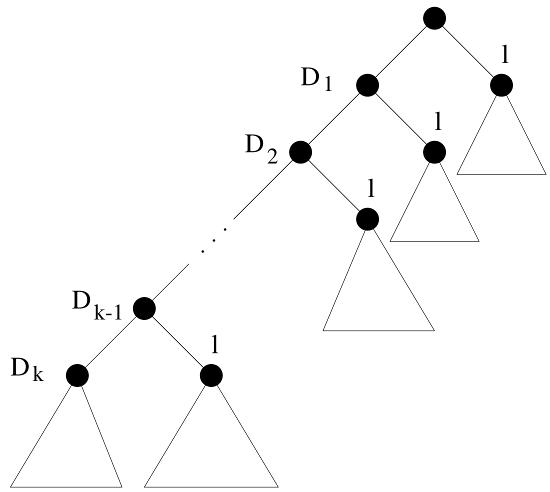

Figure 2: Inductive step for the second case

We estimate $g(\Sigma)$ by using the induction hypothesis, and the inequalities $1 \leq \log r(\Sigma \upharpoonright l) \leq \log r(\Sigma) - 1$:

$$\begin{aligned} g(\Sigma) &\leq (|\Sigma| - 1)(g(\Sigma \upharpoonright l) + 1) + g(\Sigma \upharpoonright \bar{l}) \\ &\leq (|\Sigma| - 1)(r(\Sigma \upharpoonright l) \cdot |\Sigma \upharpoonright l|^{\log r(\Sigma \upharpoonright l)} + 1) \\ &\qquad + r(\Sigma \upharpoonright \bar{l}) \cdot |\Sigma \upharpoonright \bar{l}|^{\log r(\Sigma \upharpoonright \bar{l})} \\ &\leq |\Sigma| \cdot r(\Sigma \upharpoonright l) \cdot |\Sigma|^{\log r(\Sigma) - 1} + r(\Sigma \upharpoonright \bar{l}) \cdot |\Sigma|^{\log r(\Sigma)} \\ &\leq r(\Sigma \upharpoonright l) \cdot |\Sigma|^{\log r(\Sigma)} + r(\Sigma \upharpoonright \bar{l}) \cdot |\Sigma|^{\log r(\Sigma)} \\ &= (r(\Sigma \upharpoonright l) + r(\Sigma \upharpoonright \bar{l}))|\Sigma|^{\log r(\Sigma)} \\ &= r(\Sigma) \cdot |\Sigma|^{\log r(\Sigma)}, \end{aligned}$$

completing the induction. □

COROLLARY 4.2. *Generalized clausal tableaux quasi polynomially simulates tree resolution.*

5. A LOWER BOUND FOR UNRESTRICTED TABLEAUX

The upper bound of the preceding section leaves open the possibility that it might be improved to a polynomial simulation of tree resolution by generalized clausal tableaux. In this section, we show that no such improvement is possible by giving a lower bound for such tableaux that is polynomially related to the upper bound provided by Theorem 4.1.

The generalized tableaux of the preceding section were based on binary branching. The lower bound proved below holds for an even more general formulation of analytic tableaux, where we allow arbitrary decompositions of clauses, not just binary decomposition.

If Σ is a set of clauses, then an *unrestricted tableau for* Σ is a tree in which an interior node d has associated with it a clause that is either a clause from Σ or occurs as a label on the branch ending in d. If a node is associated with a given clause C, then the children of the node are labelled with clauses $C_1, \ldots, C_k$, where $C_1, \ldots, C_k$ forms a partition of C (the root of the tree remains unlabelled). If C consists of a single literal, then the unique child of the node associated with C is labelled with C itself. A tableau for Σ is a *refutation of* Σ if every branch in the tableau is closed (i.e. contains a literal and its negation).

The lower bound employs the sets of clauses Σ_n based on the complete binary tree T_n already used in §2. If T is a labelled binary tree, and $\Sigma(T)$ the corresponding set of clauses, constructed as in §2, then we write $h(T)$ as an abbreviation for $h(\Sigma(T))$, where $h(\Sigma)$ is the unrestricted tableau complexity of Σ.

If Σ is a set of clauses, and $D \subseteq C \in \Sigma$, then we write $\Sigma[D]$ for the set of clauses resulting from Σ by replacing all clauses in Σ subsumed by D by D.

LEMMA 5.1. *If $l \in D$, where $D \subseteq C \in \Sigma$, then $h(\Sigma\restriction l) \leq h(\Sigma[D])$.*

Proof. This follows from the fact that $h(\Sigma\restriction l) = h(\Sigma[D]\restriction l) \leq h(\Sigma[D])$. □

A key feature of the sets of clauses Σ_n is that (as we already observed in §4), variables occurring higher in the tree T_n are more significant than those occurring lower down. We can make this observation precise as follows. Let l_1 and l_2 be literals occurring in the set of clauses $\Sigma(T)$, where T is a binary tree, and p_1 and p_2 the variables occurring in l_1 and l_2. We say that l_1 *dominates* l_2 if there is a branch in T so that both p_1 and p_2 label nodes in the branch, and the node labelled by p_1 is higher in the branch than the node labelled by p_2.

LEMMA 5.2. *If l_1 and l_2 are literals in the set of clauses $\Sigma(T)$, and l_1 dominates l_2, then $h(T\restriction l_1) \leq h(T\restriction l_2)$.*

Proof. If l_1 dominates l_2, then $\Sigma(T)\restriction l_1$ does not contain any clause having l_2 or $\overline{l_2}$ as a literal. Hence, $h(T\restriction l_1) = h((T\restriction l_2)\restriction l_1) \leq h(T\restriction l_2)$. □

Define $h(m, N)$ to be the minimum of $h(T)$, where T is a binary tree of depth at most m, and containing at least $2^m - N$ internal nodes.

LEMMA 5.3. *If $N_1 \leq N_2$, then $h(m, N_1) \geq h(m, N_2)$.*

Proof. This follows from the fact that if T' results from T by deleting an interior node, then $h(T) \geq h(T')$. □

LEMMA 5.4. *If $Q < 2^{p-1}$, then $h(p, Q-1) \geq h(p, Q) + h(p-1, Q-1)$.*

Proof. Let T be a binary tree of depth at most p, containing at least $2^p - Q - 1$ internal nodes, and such that $h(T) = h(p, Q-1)$. Furthermore, let us suppose that $T = U \oplus V$. Since $Q < 2^{p-1}$, it follows that neither U nor V is a one element tree.

Let C be the clause decomposed at the root of a minimal-size refutation of $\Sigma(T)$, and let $C_1, \ldots, C_k$ be the subclauses of C labelling the nodes in the refutation lying immediately below the root of the tableau. The clause C is a clause C_b associated with a branch b in T. Let l_1 be the highest literal occurring in C, l_2 the lowest literal in C (that is, to say, the variable in l_1 labels the root of T, while the variable in l_2 labels a node in b that has the leaf of b as a child).

One of the clauses C_i must contain the lowest literal l_2. Choose another clause C_j, where $i \neq j$, and let l_3 be the highest variable occurring in C_j (l_3 may or may not be l_1). Suppose that $T = U \oplus V$, and that the branch b belongs to the lefthand subtree U. Then $\Sigma(T)\restriction l_1 = \Sigma(T\restriction l_1) = \Sigma(V)$, and $\Sigma(T)\restriction l_2 = \Sigma(T')$, where T' has one fewer internal node than T. Lemma 5.1 and Lemma 5.2 imply that $h(T\restriction l_2) \leq h(\Sigma(T)[C_i])$, and $h(T\restriction l_1) \leq h(\Sigma(T)[C_j])$. Hence, $h(T) \geq h(T\restriction l_2) + h(T\restriction l_1)$. Since U contains at most $2^{p-1}-1$ internal nodes, it follows that V contains at least $2^{p-1} - Q$ internal nodes. It follows that

$$h(p, Q-1) \geq h(p, Q) + h(p-1, Q-1),$$

completing the proof of the lemma. □

LEMMA 5.5. *If $rN + 1 < 2^{m-1}$, then for all j, where $0 \leq j \leq r$,*

$$h(m+j, (r-j)N + 1) \geq N^j.$$

Proof. The lemma is proved by induction on j, from $j = 0$ to $j = r$. The induction step is proved by introducing an auxiliary variable k, and proving by downward induction on k, from $k = N$ to $k = 0$ that

$$h(m+j, (r-j)N + k + 1) \geq (N-k)N^{j-1}.$$

Assume the above inequality holds up to $j-1$, and for parameter j, for $k = k+1$ to N. We now want to prove the inequality for j and k. By Lemmas 5.3 and 5.4, and the induction hypothesis,

$$\begin{aligned} & h(m+j, (r-j)N + k + 1) \\ & \quad \geq h(m+j, (r-j)N + k + 2) \\ & \quad\quad + h(m+j-1, (r-j)N + k + 1) \\ & \quad \geq h(m+j, (r-j)N + (k+1) + 1 \\ & \quad\quad + h(m+j-1, (r-j+1)N + 1) \\ & \quad \geq (N - (k+1)) \cdot N^{j-1} + N^{j-1} \\ & \quad = (N-k) \cdot N^{j-1}, \end{aligned}$$

completing the induction step. □

THEOREM 5.6. *The unrestricted tableau complexity of Σ_n is $2^{\Theta(n^2)}$.*

Proof. In Lemma 5.5, set $m = \lceil 2n/3 \rceil$, $r = \lfloor n/3 \rfloor$, $N = 2^{\lfloor n/3 \rfloor}$, and $j = r$. Then for n sufficiently large, $rN + 1 < 2^{m-1}$, showing that $h(\Sigma_n) = h(n, 1) \geq N^r = 2^{\Omega(n^2)}$. Theorem 4.1 produces an upper bound of $2^{O(n^2)}$, so the theorem follows. □

COROLLARY 5.7. *Unrestricted clausal tableaux cannot p-simulate tree resolution.*

6. REFERENCES

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

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

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

[4] Daniel H. Greene and Donald E. Knuth. *Mathematics for the Analysis of Algorithms.* Birkhäuser, 1990. Third edition.

[5] D.W. Loveland. Mechanical theorem proving by model elimination. *Journal of the ACM*, 15:236–251, 1968.

[6] Fabio Massacci. The complexity of analytic and clausal tableaux. Forthcoming, *Theoretical Computer Science.*

[7] Fabio Massacci. Cook and Reckhow are wrong: Subexponential tableau proofs for their family of formulae. In Henri Pradé, editor, *13th European Conference on Artificial Intelligence*, pages 408–409. Morgan Kaufmann, 1998.

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

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

[10] Alasdair Urquhart. The complexity of propositional proofs. *The Bulletin of Symbolic Logic*, 1:425–467, 1995.