
Analytic Cut Trees

CARLO CELLUCCI, *Dipartimento di Studi Filosofici ed Epistemologici, Università di Roma 'La Sapienza', Via Nomentana 118, 00161 Roma, Italy. E-mail: cellucci@uniroma1.it*

Abstract

Smullyan maintains that the importance of cut-free proofs does not stem from cut elimination per se but rather from the fact that they satisfy the subformula property. In accordance with such a viewpoint in this paper we introduce analytic cut trees, a system from which cuts cannot be eliminated but satisfying the subformula property. Like tableaux analytic cut trees are a refutation system but unlike tableaux they have a single inference rule (a form of the analytic cut rule) and several branch closure rules. The main advantage of analytic cut trees over tableaux is efficiency: while analytic cut trees can simulate tableaux with an increase in complexity by at most a constant factor, tableaux cannot polynomially simulate analytic cut trees. Indeed analytic cut trees are intrinsically more efficient than any cut-free system.

Keywords: Analytic cut; subformula property; resolution

1 Introduction

In introducing his calculus of sequents Gentzen ([9] pp. 85-6) suggested that, if one were not interested in cut elimination, nine out of his fourteen operational inference rules could be replaced by axioms. Apparently he saw no way of replacing the remaining five operational inference rules, but such a restriction was removed by Smullyan through his analytic cut system [15]. Inference rules exactly corresponding to Smullyan's [15] axioms were also considered by Kneale [12].

Gentzen's ([9] p. 69) basic aim was to show that every purely logical proof can be reduced to a definite, though not unique, normal form, hence he considered cut elimination as primary and the subformula property as a byproduct. A different viewpoint was put forward by Smullyan ([15] p. 560) according to whom the real importance of cut-free proofs is not cut elimination per se but rather that such proofs satisfy the subformula property, which is all that matters for applications. In that respect it is the subformula property, not cut elimination, that is primary. Consistently with such a view, in Smullyan's [15] analytic cut system Gentzen's unrestricted cut rule is replaced by an analytic cut rule in which the cut formula is a subformula of at least one formula in the conclusion. Although cuts cannot be eliminated from the resulting system, the subformula property still holds.

As it is well known, Smullyan's [16] tableaux are a streamlined version of Kleene's [11] cut-free sequent calculus **G4** (see [2] for details). Is there a similarly streamlined counterpart of Smullyan's analytic cut system? In this paper we introduce such a counterpart in the form of analytic cut trees, i.e. trees of formulas generated by a single inference rule corresponding to an analytic cut rule. Smullyan's [15] axioms are replaced by suitable closure conditions on branches, consisting of contradiction rules depending on the meaning of logical symbols. While tableaux have several inference rules and a single closure condition, analytic cut trees have a single inference rule and

several closure conditions.

The main advantage of analytic cut trees over tableaux consists in their efficiency. Relative efficiency of proof systems can be studied in terms of the notion of p -simulation ([4], [5]): a proof system \mathbf{S} is said to p -simulate a proof system \mathbf{S}' if there exists a polynomial $p(n)$ such that, for any sentence φ , if φ has an \mathbf{S} -proof of complexity less than or equal to n , then has an \mathbf{S}' -proof of complexity less than or equal to $p(n)$. Then the computational inadequacy of tableaux is expressed by the fact that, while analytic cut trees can simulate tableaux with an increase in complexity by at most a constant factor, tableaux cannot p -simulate analytic cut trees. Indeed no cut-free system can p -simulate analytic cut trees. If one accepts the view that, whenever \mathbf{S} p -simulates \mathbf{S}' but \mathbf{S}' does not p -simulate \mathbf{S} , \mathbf{S} is essentially more efficient than \mathbf{S}' , then analytic cut trees are essentially more efficient than tableaux.

Admittedly the p -simulation criterion has some limitations. First of all, p -simulation is a relative complexity relation, not an absolute one. Moreover, it seems doubtful that a polynomial $p(n)$ of very high degree could be accepted as a bound. Finally, it seems unlikely that there are p -bounded [5] proof systems, i.e. systems \mathbf{S} for which there exists a polynomial $p(n)$ such that any theorem φ of \mathbf{S} of complexity n has a proof in \mathbf{S} of complexity less than or equal to $p(n)$. This, however, does not diminish the interest of the p -simulation criterion: on the contrary, it makes it even more important, because distinct intractable systems may differ with respect to the class of theorems for which they are p -bounded.

The subject of systems from which cuts cannot be eliminated but satisfying the subformula property is still a relatively unexplored one. Apart from Smullyan's [15] analytic cut system, D'Agostino and Mondadori's [6] analytic elimination trees seem to be the only development in this area. Analytic cut trees provide a further development.

It has been recently suggested by Oppacher and Suen [13] and Fitting [7] that tableaux provide a viable alternative to Robinson's [14] resolution as a basis for automated deduction. From the viewpoint of efficiency such a suggestion has some limitations. For, by a result of [4] tableaux cannot p -simulate resolution and, as shown in this paper, tableaux cannot p -simulate analytic cut trees. On the other hand, analytic cut trees can linearly simulate resolution.

We do not intend to claim that analytic cut trees provide a better approach to automated deduction than existing systems. In order to use them as a practical automated deduction system it would be advisable to enrich them with suitable derived rules which, while inessential for completeness, might be important for flexibility. The proper choice of such additional rules is a subject for further investigation. However analytic cut trees present analytic cut systems, as it were, in pure form and seem therefore to provide an appropriate framework for the theoretical study of this class of systems.

2 First order languages

First order languages are defined in the usual way except that for conciseness we use a variant of Smullyan's [16] uniform notation.

Definition 2.1 1. We consider first order languages \mathbf{L} containing variables, constants, n -ary predicate symbols ($n \geq 0$), logical symbols $\neg, \wedge, \vee, \rightarrow, \forall, \exists$.

2. We denote variables by letters x, y, z, \dots , constants by letters k, l, m, \dots , predicate symbols by letters P, Q, R, \dots (where the arity of predicate symbols will always be clear from the context). All such letters may have subscripts.
3. The result of substituting a constant k for every occurrence of a variable x in a formula φ will be denoted by $\varphi[x/k]$.

Definition 2.2 1. Let \mathbf{L} be a first order language. We expand \mathbf{L} adding new constants, called parameters.

2. Parameters will be denoted by letters a, b, \dots .
3. Let \mathbf{L}^p denote the language \mathbf{L} expanded with parameters.

Definition 2.3 1. We group all formulas of one of the forms $(\varphi \circ \psi)$ and $\neg(\varphi \circ \psi)$, where \circ is among the symbols $\wedge, \vee, \rightarrow$ into two categories, α formulas and β formulas, and define, for each α formula, two components which we denote by α_1 and α_2 , and for each formula β , two components which we denote by β_1 and β_2 , as shown by the following table:

α	α_1	α_2	β	β_1	β_2
$\varphi \wedge \psi$	φ	ψ	$\neg(\varphi \wedge \psi)$	$\neg\varphi$	$\neg\psi$
$\neg(\varphi \vee \psi)$	$\neg\varphi$	$\neg\psi$	$\varphi \vee \psi$	φ	ψ
$\neg(\varphi \rightarrow \psi)$	φ	$\neg\psi$	$\varphi \rightarrow \psi$	$\neg\varphi$	ψ

2. We group all formulas of one of the forms $Qx\varphi$ and $\neg Qx\varphi$, where $Q \in \{\forall, \exists\}$, into two categories, γ formulas, and δ formulas, and define, for each γ formula, its instances $\gamma(a)$, and for each δ formula, its instances $\delta(a)$, as shown by the following table:

γ	$\gamma(a)$	δ	$\delta(a)$
$\forall x\varphi$	$\varphi[x/a]$	$\exists x\varphi$	$\varphi[x/a]$
$\neg\exists x\varphi$	$\neg\varphi[x/a]$	$\neg\forall x\varphi$	$\neg\varphi[x/a]$

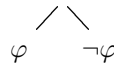
Definition 2.4 The complement of a formula φ , written $\overline{\varphi}$, is defined as follows:

1. $\overline{\varphi} = \psi$ if $\varphi = \neg\psi$
2. $\overline{\varphi} = \neg\varphi$ if φ is not a negation.

3 Analytic cut trees

In this section we state the rules of analytic cut trees. As anticipated, the system consists of a single inference rule, i.e. the analytic cut rule, but has several closure rules.

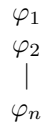
Definition 3.1 We introduce a rule which allows to turn a tree with sentences as node labels into another such a tree. Suppose we have a finite tree \mathbf{T} (written with the root at the top) with nodes labeled by formulas of \mathbf{L}^p . Select a branch \mathcal{B} of \mathbf{T} and a sentence φ of \mathbf{L}^p , lengthen \mathcal{B} adding left and right children to the final node of \mathcal{B} and label one φ and the other $\neg\varphi$. Call the resulting tree \mathbf{T}' . We say that \mathbf{T}' results from \mathbf{T} by an application of the cut rule (with cut formula φ). Schematically the cut rule can be written as follows:



Remark 3.2 Reading the splitting of a branch \mathcal{B} of \mathbf{T} into two branches disjunctively, the above cut rule corresponds to the excluded middle principle.

Definition 3.3 Let $\Gamma = \{\varphi_1, \dots, \varphi_n\}$ be a finite set of sentences of \mathbf{L} . The notion of analytic cut tree for Γ is defined as follows.

1. The one branch tree whose nodes are labeled by the members of Γ ,



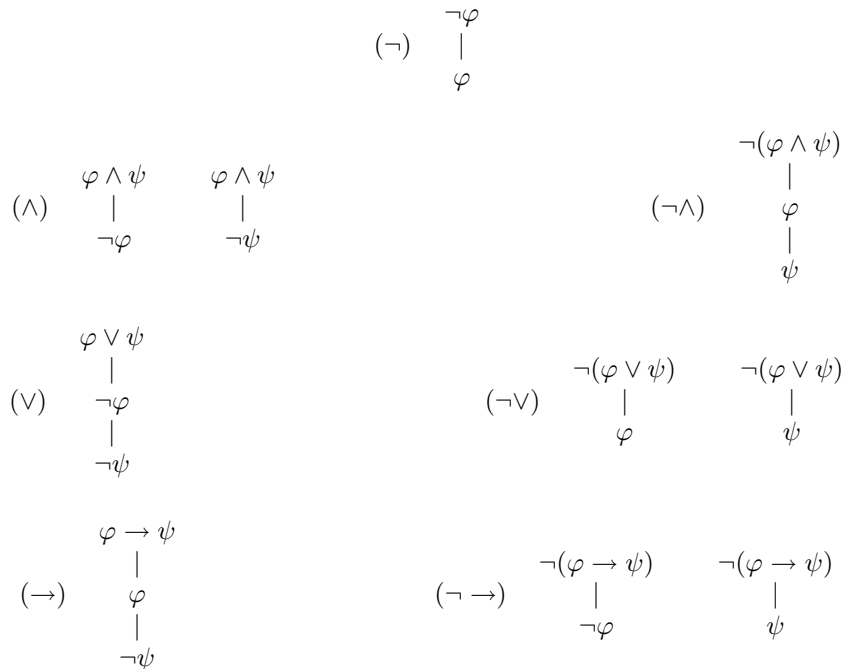
is an analytic cut tree for Γ .

2. If \mathbf{T} is an analytic cut tree for Γ and \mathbf{T}' results from \mathbf{T} by an application of the cut rule with cut formula any sentence that is a subformula or the negation of a subformula of some sentence of Γ , then \mathbf{T}' is an analytic cut tree for Γ .

Remark 3.4 1. An analytic cut tree for a set of sentences Γ of \mathbf{L} may contain sentences of \mathbf{L}^p .

2. In an analytic cut tree for Γ all applications of the cut rule are restricted to be analytic in the sense that their cut formula must be a subformula or the negation of a subformula of some sentence of Γ .

Definition 3.5 The closure rules for analytic cut trees are the following:



$$(\forall) \quad \begin{array}{c} \forall x\varphi \\ | \\ \neg\varphi[x/a] \end{array}$$

$$(\neg\forall) \quad \begin{array}{c} \neg\forall x\varphi \\ | \\ \varphi[x/a] \\ \text{with proviso} \\ \text{(see below)} \end{array}$$

$$(\exists) \quad \begin{array}{c} \exists x\varphi \\ | \\ \neg\varphi[x/a] \\ \text{with proviso} \\ \text{(see below)} \end{array}$$

$$(\neg\exists) \quad \begin{array}{c} \neg\exists x\varphi \\ | \\ \varphi[x/a] \end{array}$$

In $(\neg\forall)$ and (\exists) the parameter a , called its proper parameter, is subject to the condition that a must be new to the given branch and must not be the proper parameter of any other application of $(\neg\forall)$ or (\exists) in any other branches.

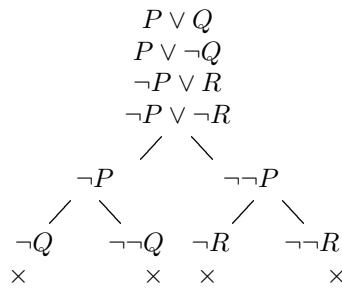
Definition 3.6 1. We say that a branch \mathcal{B} of an analytic cut tree is closed if \mathcal{B} contains formulas of the form indicated in one of the closure rules for analytic cut trees; open otherwise.

2. We say that an analytic cut tree \mathbf{T} is closed if all branches \mathcal{B} of \mathbf{T} are closed.

Definition 3.7 1. Let φ be a sentence of \mathbf{L} . A proof of φ in the analytic cut tree system is a closed analytic cut tree for $\{\neg\varphi\}$.

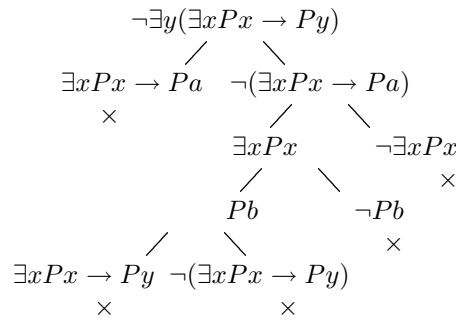
2. We say that φ is a theorem of the analytic cut tree system if there exists a proof of φ in that system.

Example 3.8 The following is a closed analytic cut tree for the set of sentences $\{P \vee Q, P \vee \neg Q, \neg P \vee R, \neg P \vee \neg R\}$:

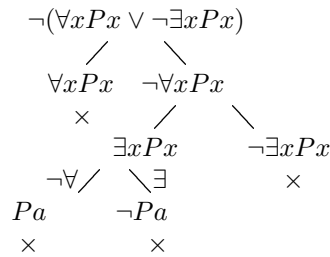


Here and in what follows we express that a branch is closed by writing \times under its final node.

Example 3.9 The following is a closed analytic cut tree for $\{\neg\exists y(\exists xPx \rightarrow Py)\}$:



Remark 3.10 *The proviso that in an application of $(\neg\forall)$ or (\exists) the parameter a , not only must be new to the given branch, but must not be the proper parameter of any other application of $(\neg\forall)$ or (\exists) in other branches, is essential for soundness. E.g., if the second half of the proviso were relaxed, one could construct the following 'proof' of the invalid sentence $\forall xPx \vee \neg\exists xPx$ in the analytic cut tree system:*



4 A complete proof procedure

In this section we describe a complete proof procedure for analytic cut trees. For brevity we use the uniform notation.

Definition 4.1 *In terms of the uniform notation analytic cut trees can be expressed as follows. The cut rule takes the form:*

$$(Cut) \quad \begin{array}{c} \wedge \\ \varphi \quad \bar{\varphi} \end{array}$$

The closure rules take the form:

$$(\neg) \quad \begin{array}{c} \neg\varphi \\ | \\ \varphi \end{array}$$

$$(\alpha) \quad \begin{array}{c} \alpha \\ | \\ \bar{\alpha}_1 \end{array} \quad \begin{array}{c} \alpha \\ | \\ \bar{\alpha}_2 \end{array}$$

$$(\beta) \quad \begin{array}{c} \beta \\ | \\ \bar{\beta}_1 \\ | \\ \bar{\beta}_2 \end{array}$$

$$(\gamma) \frac{\gamma}{\gamma(a)}$$

$$(\delta) \frac{\delta}{\delta(a)}$$

with proviso
(see below)

In (δ) the parameter a , called its proper parameter, is subject to the condition that a must be new to the given branch and must not be the proper parameter of any other application of (δ) in any other branches.

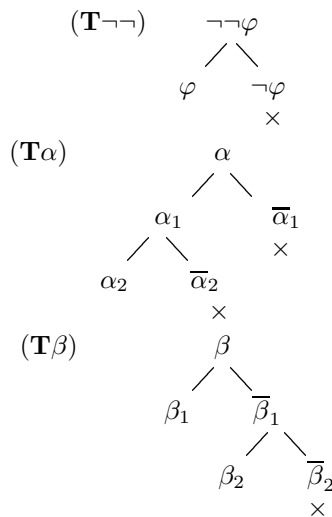
Definition 4.2 We say that a branch \mathcal{B} of an analytic cut tree is complete if it satisfies the following conditions:

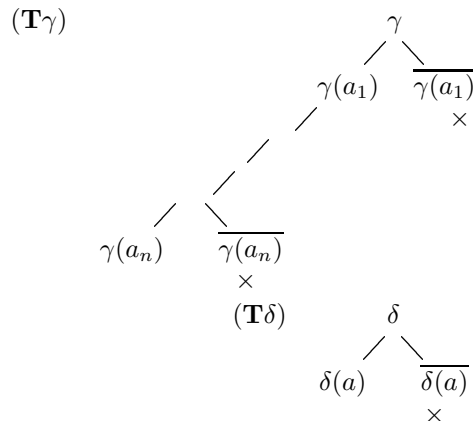
1. $\neg\neg\varphi \in \mathcal{B} \Rightarrow \varphi \in \mathcal{B}$;
2. $\alpha \in \mathcal{B} \Rightarrow \alpha_1 \in \mathcal{B}$ and $\alpha_2 \in \mathcal{B}$;
3. $\beta \in \mathcal{B} \Rightarrow \beta_1 \in \mathcal{B}$ or $\beta_2 \in \mathcal{B}$;
4. $\gamma \in \mathcal{B} \Rightarrow \gamma(a) \in \mathcal{B}$ for every parameter a occurring in some sentence in \mathcal{B} , and for at least one parameter a ;
5. $\delta \in \mathcal{B} \Rightarrow \delta(a) \in \mathcal{B}$ for at least one parameter a .

Definition 4.3 We say that a branch \mathcal{B} of an analytic cut tree is a Hintikka branch if it is complete and satisfies the following additional condition:

6. For no atomic formula P , both $P \in \mathcal{B}$ and $\neg P \in \mathcal{B}$.

Definition 4.4 We call basic construction trees the following trees:





Theorem 4.5 *There exists an algorithm that, applied to any logically valid sentence φ , yields a proof of φ in the analytic cut tree system.*

PROOF. The algorithm can be described as follows. Start the analytic cut tree by placing $\neg\varphi$ at the origin. This concludes the stage 1 of our procedure.

Now suppose we have concluded the stage n . Then the stage $n + 1$ is determined as follows. If the analytic cut tree already at hand is closed, then STOP. If the analytic cut tree already at hand has a complete open branch, then STOP. Else, pick up an unchecked sentence ψ which is on at least one open branch and extend the analytic cut tree at hand as follows. Consider every open branch that contains the node ψ and

1) if ψ is of one of the forms $\neg\neg\varphi$, α , β , then lengthen the branch by adding the basic construction tree (**T** $\neg\neg$), (**T** α), (**T** β) respectively, at its end, and check the node ψ ;

2) if ψ is of the form δ , then lengthen the branch by adding the basic construction tree (**T** δ) at its end, using a parameter that does not occur in the analytic cut tree as a , and check the node ψ ;

3) if ψ is of the form γ , then lengthen the branch by adding the basic construction tree (**T** γ) at its end, using all parameters occurring in the branch as a_1, \dots, a_n (or a given parameter a if no parameter occurs in the branch), and repeat the procedure for every sentence of the form γ occurring in the analytic cut tree at hand.

Having performed acts 1) - 3), this concludes the stage $n + 1$ of our procedure.

Now, suppose that φ is logically valid but the analytic cut tree **T** for $\{\neg\varphi\}$ produced by the algorithm is not closed. Clearly there are two ways in which **T** might fail to be closed: a) **T** has at least one finite open branch \mathcal{B} that is complete; b) **T** is infinite. In case a), since \mathcal{B} is open, \mathcal{B} is a Hintikka branch. In case b), by König's lemma **T** must have an infinite open branch \mathcal{B} . By the definition of the algorithm \mathcal{B} must be complete, thus since \mathcal{B} is open it must be a Hintikka branch. Since in both cases \mathcal{B} is a Hintikka branch, it is satisfiable (see e.g. [16] p. 58). But $\neg\varphi$ is the origin of **T**, so $\neg\varphi \in \mathcal{B}$, hence $\{\neg\varphi\}$ is satisfiable. This contradicts the fact that φ is logically valid. ■

5 Relationships of analytic cut trees to tableaux

In this section we show that analytic cut trees can p -simulate tableaux, hence in the worst case closed analytic cut trees are not essentially longer than the corresponding

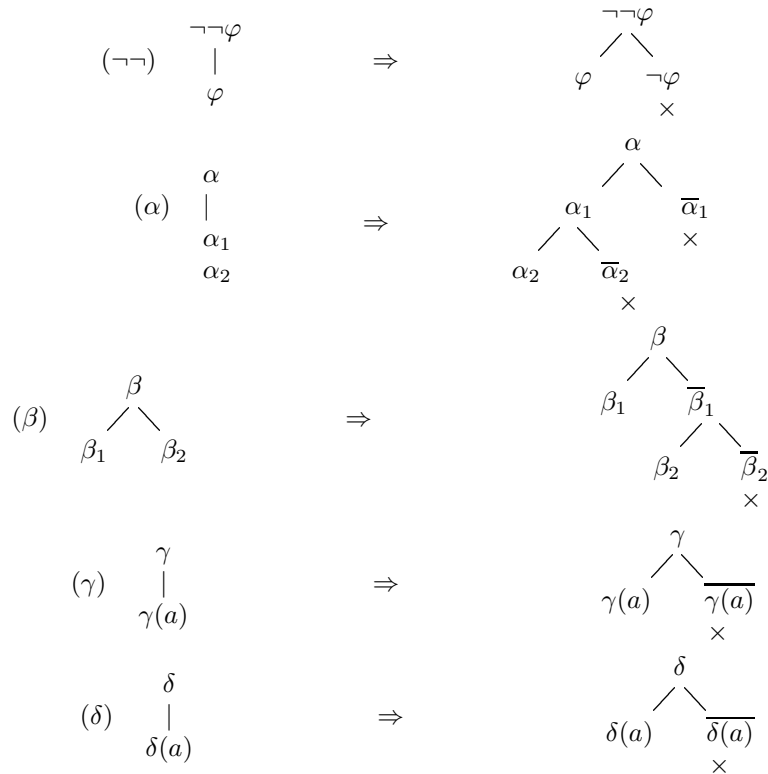
closed tableaux. For background on tableaux see Smullyan [16].

- Definition 5.1** 1. The complexity of an analytic cut tree \mathbf{T} for Γ , denoted by $|\mathbf{T}|$, is the number of nodes of \mathbf{T} , except that all the initial nodes labeled by the members of Γ are counted as a single node.
2. The complexity of a tableau \mathbf{T} for Γ , denoted by $|\mathbf{T}|$, is the number of nodes of \mathbf{T} , except that all the initial nodes labeled by the members of Γ are counted as a single node.

Remark 5.2 Taking the number of nodes as a complexity measure, while adequate for negative results about the p -simulation relation, is generally inadequate for positive results. However it may be adequate also for positive results whenever the length (total number of symbols) of sentences is not significantly increased by the p -simulation procedure under consideration. Since all of the p -simulation procedures considered here will be of such a kind, in what follows we will confine ourselves to this simple complexity measure.

Theorem 5.3 There exists an algorithm which converts every closed tableau \mathbf{T} for Γ into a closed analytic cut tree \mathbf{T}' for Γ such that $|\mathbf{T}'| \leq 3|\mathbf{T}|$.

PROOF. Tableau inference rules can be simulated by analytic cut trees as follows:



Now, if \mathbf{T} is a closed tableau for Γ , then replace each application of a tableau rule in \mathbf{T} with its simulation by an analytic cut tree. The result \mathbf{T}' is a closed analytic

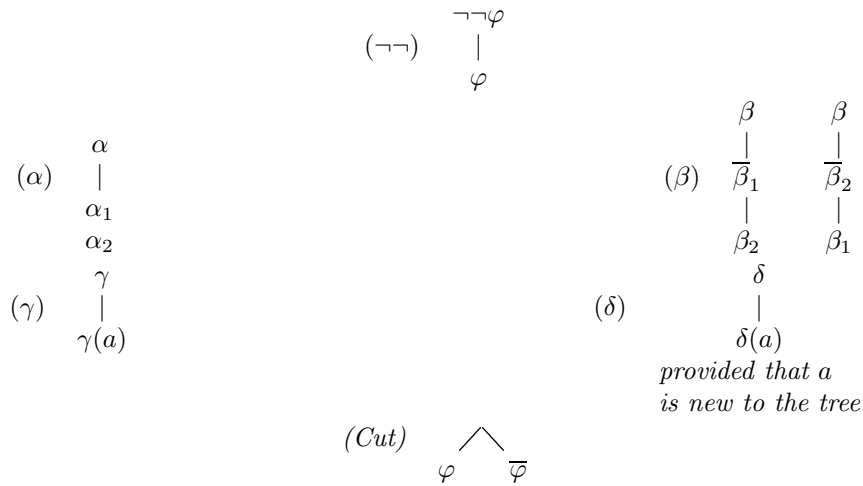
cut tree for Γ such that $|\mathbf{T}'| \leq |\mathbf{T}| + 2|\mathbf{T}| = 3|\mathbf{T}|$ since 2 is the maximum number of additional nodes generated by the simulation of a tableau rule by an analytic cut tree. ■

Remark 5.4 *By Theorem 5.3 analytic cut trees, not only can p -simulate tableaux, but can also linearly simulate them, in the sense that the function $p(n)$ has the form $k \cdot n$ for some given constant k .*

6 Relationships of analytic cut trees to analytic elimination trees

In this section we show that analytic cut trees and analytic elimination trees can p -simulate each other, hence they are of comparable efficiency. For background on analytic elimination trees see D’Agostino and Mondadori [6].

Definition 6.1 *The rules of analytic elimination trees are the following:*



Definition 6.2 *Let Γ be a finite set of sentences of \mathbf{L} . The notion of analytic elimination tree for Γ is defined as follows.*

1. *The one branch tree whose nodes are labeled by the members of Γ is an analytic elimination tree for Γ .*
2. *If \mathbf{T} is an analytic elimination tree for Γ and \mathbf{T}' results from \mathbf{T} by an application of one of the rules $(\neg\neg), (\alpha), (\beta), (\gamma), (\delta)$, then \mathbf{T}' is an analytic elimination tree for Γ .*
3. *If \mathbf{T} is an analytic elimination tree for Γ and \mathbf{T}' results from \mathbf{T} by an application of the cut rule with cut formula any sentence which is a subformula or the negation of a subformula of some sentence of Γ , then \mathbf{T}' is an analytic elimination tree for Γ .*

Definition 6.3 1. *A branch \mathcal{B} of an analytic elimination tree for Γ is said to be closed if both ρ and $\neg\rho$ occur on \mathcal{B} , for some atom ρ .*

6. RELATIONSHIPS OF ANALYTIC CUT TREES TO ANALYTIC ELIMINATION TREES 743

2. An analytic elimination tree \mathbf{T} for Γ is said to be closed if every branch of \mathbf{T} is closed.

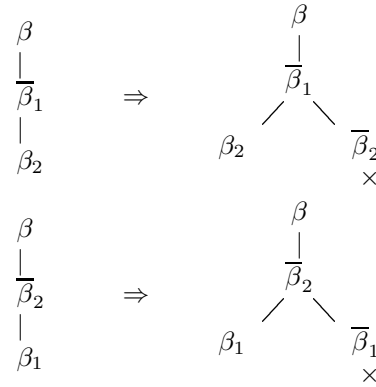
Definition 6.4 1. A proof of φ in the analytic elimination tree system is a closed analytic elimination tree for $\{\neg\psi\}$.

2. We say that φ is a theorem of the analytic elimination tree system if there exists a proof of φ in that system.

Definition 6.5 By the complexity of an analytic elimination tree \mathbf{T} for Γ , denoted by $|\mathbf{T}|$, we mean the number of nodes of \mathbf{T} , except that all the initial nodes labeled by the members of Γ are counted as a single node.

Theorem 6.6 There exists an algorithm which converts every closed analytic elimination tree \mathbf{T} for Γ into a closed analytic cut tree \mathbf{T}' for Γ such that $|\mathbf{T}'| \leq 3|\mathbf{T}|$.

PROOF. Rules $(\neg\neg)$, (α) , (γ) , (δ) of analytic elimination trees can be simulated by analytic cut trees as in the proof of Theorem 5.3. On the other hand rule (β) , which is of different form, can be simulated as follows:



Now, if \mathbf{T} is a closed analytic elimination tree for Γ , then replace each application of a rule of analytic elimination trees with its simulation by analytic cut trees. Applications of the analytic cut rule are left unchanged. The result \mathbf{T}' is a closed analytic cut tree for Γ such that $|\mathbf{T}'| \leq |\mathbf{T}| + 2|\mathbf{T}| = 3|\mathbf{T}|$ since 2 is the maximum number of additional nodes generated by the simulation of a rule of analytic elimination trees by an analytic cut tree. ■

Theorem 6.7 There exists an algorithm which converts every closed analytic cut tree \mathbf{T} for Γ into a closed analytic elimination tree \mathbf{T} for Γ such that $|\mathbf{T}'| \leq 3|\mathbf{T}|$.

PROOF. The closure rules for analytic cut trees can be simulated in the following way. Rule (\neg) is just the closure rule for analytic elimination trees. Rules (α) , (β) , (γ) , (δ) can be simulated as follows:



$$\begin{array}{ccc}
 (\alpha) \quad \begin{array}{c} \alpha \\ | \\ \overline{\alpha_2} \end{array} & \Rightarrow & \begin{array}{c} \alpha \\ | \\ \alpha_1 \\ \alpha_2 \\ | \\ \overline{\alpha_2} \\ \beta \\ | \\ \overline{\beta_1} \\ | \\ \beta_2 \\ | \\ \overline{\beta_2} \end{array} \\
 (\beta) \quad \begin{array}{c} \beta \\ | \\ \overline{\beta_1} \\ | \\ \overline{\beta_2} \end{array} & \Rightarrow & \begin{array}{c} \gamma \\ | \\ \gamma(a) \\ | \\ \overline{\gamma(a)} \\ \delta \\ | \\ \delta(a) \\ | \\ \overline{\delta(a)} \end{array} \\
 (\gamma) \quad \begin{array}{c} \gamma \\ | \\ \overline{\gamma(a)} \end{array} & \Rightarrow & \\
 (\delta) \quad \begin{array}{c} \delta \\ | \\ \overline{\delta(a)} \end{array} & \Rightarrow &
 \end{array}$$

Now, if \mathbf{T} is a closed analytic cut tree for Γ , then replace each application of a closure rule for analytic cut trees with its simulation by an analytic elimination tree. Applications of the analytic cut rule are left unchanged. The result \mathbf{T}' is a closed analytic elimination tree such that $|\mathbf{T}'| \leq |\mathbf{T}| + 2|\mathbf{T}| = 3|\mathbf{T}|$ since 2 is the maximum number of additional nodes generated by the simulation of a closure rule for analytic cut trees by an analytic elimination tree. ■

7 Computational inefficiency of tableaux

In this section we show that tableaux cannot p -simulate analytic cut trees, hence the latter are essentially more efficient than the former. We also discuss the source of the computational inefficiency of tableaux.

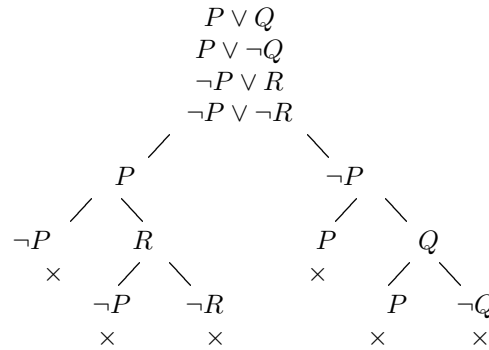
Theorem 7.1 *There exists no polynomial $p(n)$ such that, if there is a closed analytic cut tree \mathbf{T} for Γ with $|\mathbf{T}| \leq n$, then there is a closed tableau \mathbf{T}' for Γ with $|\mathbf{T}'| \leq p(n)$.*

PROOF. By [6] there exists no polynomial $p(n)$ such that, if there is a closed analytic elimination tree \mathbf{T} for Γ with $|\mathbf{T}| \leq n$, then there is a closed tableau \mathbf{T}' for Γ with $|\mathbf{T}'| \leq p(n)$. From this together with Theorem 6.7 we obtain the result. ■

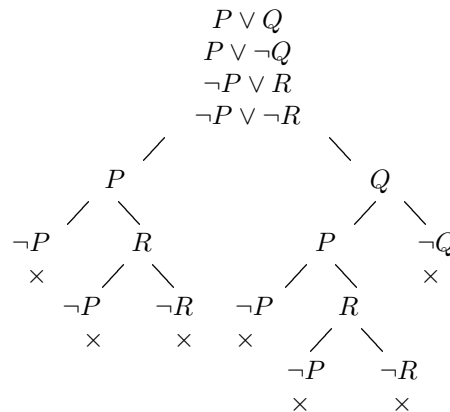
Remark 7.2 *In view of the fact that by Remark 5.4 analytic cut trees can linearly simulate tableaux, using Theorem 7.1 we may conclude that analytic cut trees, not only are essentially more efficient than tableaux, but are also uniformly more efficient than the latter.*

Remark 7.3 *The fact that tableaux cannot p -simulate analytic cut trees contradicts Beth's ([1] p. 23) original motivation in introducing tableaux, that they would allow to construct proofs which were remarkably concise and could even be proved to be the shortest possible ones.*

Example 7.4 *The inefficiency of tableaux can be ascribed to the fact that they generally involve an unnecessary duplication of subtrees. For example, if we expand the tableau rules by adding the analytic cut rule, then we have the following closed tableau for $\{P \vee Q, P \vee \neg Q, \neg P \vee R, \neg P \vee \neg R\}$:*



On the other hand if we confine ourselves to the cut-free tableau rules we may only construct the following closed tableau which contains two duplicate subtrees:



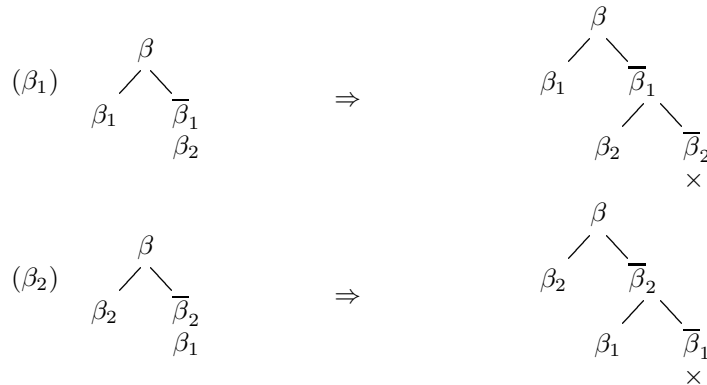
Remark 7.5 *In a sense the above example contradicts Gentzen's ([9] p. 69) claim that the essential property of cut-free proofs is that they are not roundabout. Such claim is plausible if, as Gentzen ([9] pp. 69, 88) suggests, by a roundabout proof we mean one into which no concepts enter other than those occurring in its final result. On the other hand, as Example 7.4 shows, the claim in question is implausible if by a roundabout proof we mean one which contains no redundancies. That the inefficiency of tableaux stems from an unnecessary duplication of subtrees in tableaux is supported by Toledo's ([17] Appendix 1) result that any closed tableau \mathbf{T} for Γ containing cuts can be converted into a closed cut-free tableau \mathbf{T}' for Γ such that, if in calculating the complexity of tableaux we ignore duplicated subtrees, then the complexity of \mathbf{T}' is*

actually smaller than that of \mathbf{T} , and smaller by at least the number of cut formulas, negations of cut formulas and their descendants in \mathbf{T} .

Remark 7.6 A way of improving the efficiency of tableaux would be to add the analytic cut rule to the tableau rules, but such an addition would clearly be ad hoc because the cut-free tableau rules are already complete. A more satisfactory solution is to assign analytic cut a role in the enumeration of the possible cases. This corresponds to replacing the standard symmetric tableau rule (β) with the following two asymmetric rules:



By ([6] Corollary 5.13) cut-free tableaux cannot p -simulate tableaux in which the asymmetric (β_1) and (β_2) rules are allowed, hence the latter are essentially more efficient than the former. On the other hand analytic cut trees can linearly simulate the asymmetric rules as follows:



Remark 7.7 Efficiency is not the only weak spot of tableaux. Contrary to Beth's ([1] p. 21) suggestion that tableaux exactly reflect the classical meaning of logical operations, it can be argued that they do not capture the essentials of classical semantics. Indeed, there is a 3-valued semantics, specifically the one defined in Girard ([10] pp. 168-169), in which all tableau rules are valid but the cut rule is not valid.

8 Relationship of analytic cut trees to resolution

In this section we show that sequent analytic cut trees can p -simulate resolution. For background on resolution see Chang and Lee [3] or Gallier [8].

- Definition 8.1**
1. We expand formulas by adding formulas of the form $\varphi_1 \vee \dots \vee \varphi_n$, for any finite (possibly empty) list of formulas $\varphi_1, \dots, \varphi_n$. We call $\varphi_1 \vee \dots \vee \varphi_n$ the generalized disjunction of $\varphi_1, \dots, \varphi_n$.
 2. For $n \geq 2$ we identify $\varphi_1 \vee \dots \vee \varphi_n$ with $(\dots(\varphi_1 \vee \varphi_2)\dots) \vee \varphi_n$.
 3. For $n = 1$ we identify $\varphi_1 \vee \dots \vee \varphi_n$ with φ_1 .

4. For $n = 0$ we denote $\varphi_1 \vee \dots \vee \varphi_n$ by \square .

Remark 8.2 Since disjunction is both commutative and associative, in a disjunction of many items placement of parentheses and order does not matter. In view of this, expanding formulas by adding generalized disjunctions is justified.

Definition 8.3 We expand the closure rules for analytic cut trees by adding the following new closure rule for generalized disjunction which is a generalized version of (\vee):

$$\begin{array}{ccc}
 & \varphi_1 \vee \dots \vee \varphi_n & \\
 & | & \\
 (\vee) & \neg\varphi_1 & (n \geq 2) \\
 & \vdots & \\
 & \neg\varphi_n &
 \end{array}$$

Definition 8.4 1. A literal is an atom or the negation of an atom.

2. A clause is a generalized disjunction $\varphi_1 \vee \dots \vee \varphi_n$ in which each member φ_i is a literal.

3. We call \square the empty clause.

4. A clause containing no quantifiers and no variables is called a ground clause.

Definition 8.5 1. Let χ_1 and χ_2 be two ground clauses with ρ occurring as a member of χ_1 and $\neg\rho$ occurring as a member of χ_2 . Let χ be the result of deleting all occurrences of ρ from χ_1 , deleting all occurrences of $\neg\rho$ from χ_2 and then combining the two resulting ground clauses into a single one. We say that χ is obtained from χ_1 and χ_2 by the ground resolution rule.

2. Let Γ be a finite set of ground clauses. A ground resolution derivation from Γ is a finite tree (written with the root at the bottom) labeled with ground clauses such that if node ν is labeled with χ then: if ν is a leaf node, χ must be a member of Γ ; if ν has children, their labels must be the premises from which χ is obtained by the ground resolution rule. If the root of the tree is labeled with χ , then we say that the derivation is a ground resolution derivation of χ from Γ .

3. A ground resolution derivation of \square from Γ is called a ground resolution refutation for Γ .

Remark 8.6 Without loss of generality we may confine ourselves to considering ground resolution refutations instead of resolution refutations for general clauses. This follows from the Lifting Lemma (see e.g. Gallier [8] pp. 400-403) by which, if Γ is a finite set of clauses and Δ is a finite set of ground instances of Γ , then any ground resolution refutation \mathbf{T} for Δ can be converted into a homomorphic resolution refutation \mathbf{T}' for Γ .

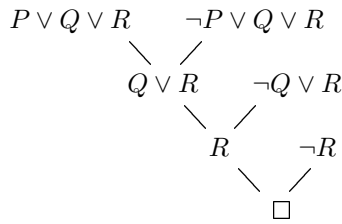
Theorem 8.7 There exists an algorithm which converts every ground resolution refutation \mathbf{T} for Γ into a closed analytic cut tree \mathbf{T}' for Γ (with the additional closure rule) such that $|\mathbf{T}'| = |\mathbf{T}|$.

PROOF. A procedure for converting every ground resolution refutation \mathbf{T} into a closed analytic cut tree (with the additional rules) can be described as follows.

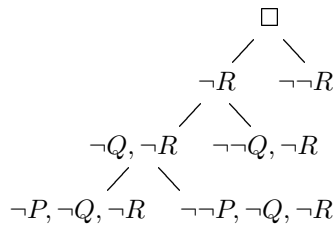
1. For each node ν of \mathbf{T} , replace the ground clause $\varphi_1 \vee \dots \vee \varphi_n$ labeling ν by the set of sentences $\{\neg\varphi_1, \dots, \neg\varphi_n\}$ consisting of the negations of its members. The clause labeling the root of \mathbf{T} , i.e. \square , is left unchanged. Then turn the tree upside down. Let \mathbf{T}° be the resulting tree.
2. For each node ν of \mathbf{T}° except the root, replace the set of sentences labeling ν by the sentence occurring in it but not in the set of sentences labeling the node standing immediately above ν in the branch. Then replace the clause \square labeling the root by the one branch tree whose nodes are labeled by the members of Γ .

The resulting tree \mathbf{T}' is an analytic cut tree because every application of the ground resolution rule in \mathbf{T} is converted in \mathbf{T}' into an application of the cut rule with cut formula a literal which is a subformula of Γ . Moreover, it is easy to see that \mathbf{T}' is closed. For, if $\{\neg\varphi_1, \dots, \neg\varphi_n\}$ is a set of sentences labeling any leaf ν of \mathbf{T}° , \mathcal{B} is the branch of \mathbf{T}° to which ν belongs, and \mathcal{B}' is the branch of \mathbf{T}' corresponding to \mathcal{B} , then by construction each of the sentences $\varphi_1 \vee \dots \vee \varphi_n, \neg\varphi_1, \dots, \neg\varphi_n$ labels a node of \mathcal{B}' , hence \mathcal{B}' is closed. Finally, trivially $|\mathbf{T}| = |\mathbf{T}^\circ| = |\mathbf{T}'|$. ■

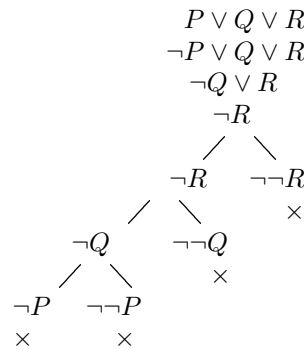
Example 8.8 Consider the following ground resolution refutation \mathbf{T} for $\Gamma = \{P \vee Q \vee R, \neg P \vee Q \vee R, \neg Q \vee R, \neg R\}$:



If, for each node ν of \mathbf{T} , we replace the ground clause $\varphi_1 \vee \dots \vee \varphi_n$ labeling ν by the set of sentences $\{\neg\varphi_1, \dots, \neg\varphi_n\}$ and turn the tree upside down, we obtain the following tree \mathbf{T}° :



Now if, for each node ν of \mathbf{T}° except the root, we replace the set of sentences labeling ν by the sentence occurring in it but not in the set of sentences labeling the node standing immediately above ν in the branch, and replace the clause \square labeling the root by the one branch tree whose nodes are labeled by the members of Γ , we obtain the following closed analytic cut tree \mathbf{T}' (with the additional closure rule):



Remark 8.9 By Theorem 8.7 analytic cut trees not only can p -simulate ground resolution and hence by Remark 8.6 can p -simulate resolution, but can also linearly simulate it.

Remark 8.10 If, as in Gallier ([8] p. 118), a clause $\varphi_1 \vee \dots \vee \varphi_n$ is viewed as the set $\{\varphi_1, \dots, \varphi_n\}$ of its literals, then resolution can be considered as an analytic cut system, albeit restricted to clauses.

Acknowledgements

Research for this paper was partially supported by CNR (Consiglio Nazionale delle Ricerche), grant 92.00477.CT12.

References

- [1] Beth, E.W., Semantic entailment and formal derivability. *Medelingen der Koninklijke Nederlandse Akademie van Wetenschappen* **18** (1955), 309-342.
- [2] Cellucci, C., Using full first order logic as a programming language. *Rendiconti del Seminario Matematico Università e Politecnico di Torino*, Fascicolo speciale 1987, pp. 115-152.
- [3] Chang, C.L. and Lee, R.C.T., *Symbolic logic and mechanical theorem proving*. Boston (Academic Press) 1973.
- [4] Cook, S.A. and Reckhow, R., On the length of proofs in the propositional calculus. In *Proceedings of the 6th Annual Symposium on the Theory of Computing*, 1974, pp. 135-148.
- [5] Cook, S.A. and Reckhow, R.A., The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic* **44** (1979), 36-50.
- [6] D'Agostino, M. and Mondadori, M., The taming of the cut. Classical refutations with analytic cut. *Journal of Logic and Computation* **4** (1994), 285-319.
- [7] Fitting, M., *First-Order Logic and Automated Theorem Proving*. Berlin (Springer-Verlag) (1990).
- [8] Gallier, J.H., *Logic for Computer Science: Foundations of Automatic Theorem Proving*. New York (Harper & Row) 1986.
- [9] Gentzen, G., Investigations into logical deduction. In Szabo, M.E. (Ed.), *The Collected Papers of Gerhard Gentzen*. Amsterdam (North-Holland) 1969, pp. 68-131.
- [10] Girard, J.-Y., *Proof theory and logical complexity*. Naples (Bibliopolis) 1987.
- [11] Kleene, S.C., *Mathematical logic*. New York (Wiley) 1967.
- [12] Kneale, W., The province of logic. In Lewis, H.D. (Ed.), *Contemporary British Philosophy*. London (Allen & Unwin) 1956, pp. 237-261.
- [13] Oppacher, F. and Suen, E., HARP: A tableau-based theorem prover. *Journal of Automated Reasoning* **4** (1988), 69-100.

- [14] Robinson, J.A., A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery* **12** (1965), 23-41.
- [15] Smullyan, R.M., Analytic cut. *The Journal of Symbolic Logic* **33** (1968), 560-564.
- [16] Smullyan, R.M., *First-Order Logic*. Berlin (Springer-Verlag) 1968.
- [17] Toledo, S., *Tableau systems for first order number theory and certain higher order theories*. LNM 447, Berlin (Springer-Verlag) 1975.