Equal Rights for the Cut: Computable Non-analytic Cuts in Cut-based Proofs

MARCELO FINGER, *University of Sao Paulo, Brazil.*
E-mail: mfinger@ime.usp.br

DOV GABBAY, *King's College, London. E-mail: dg@dcs.kcl.ac.uk*

Abstract

This work studies the structure of proofs containing non-analytic cuts in the cut-based system, a sequent inference system in which the cut rule is not eliminable and the only branching rule is the cut. Such sequent system is invertible, leading to the KE-tableau decision method. We study the structure of such proofs, proving the existence of a normal form for them in the form of a comb-tree proof.

We then concentrate on the problem of efficiently computing non-analytic cuts. For that, we study the generalisation of techniques present in many modern theorem provers, namely the techniques of conflict-driven formula learning.

Keywords: Proof Theory, non-analytic cuts, sequent calculus, tableaux.

1 Introduction

Cut elimination is one of the best known and best studied proof theoretical properties of sequent-based inference systems. Its importance and centrality cannot be overemphasised. It provides a *normal form* for sequent proofs, namely cut-free proofs, which have the *subformula property*. That is, any formula occurring in the proof is a subformula of the sequent at the root of the proof. As can easily be seen in the cut rule,

$$\frac{\Gamma_1 \vdash \Delta_1, A \quad A, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ (\text{Cut})$$

the *cut formula* A is not necessarily a subformula of Γ_i or Δ_i, $i=1,2$. Systems with the subformula property are *analytic*, in the sense that no formula enters the proof that is not a subformula of the final sequent. This implies the impossibility of inferring the empty sequent ($\varnothing \vdash \varnothing$) in the logic, which is taken to be the definition of the *consistency* of a sequent calculus.

Since the seminal work of Gentzen on sequent calculi [Sza69], the presence of the cut rule in a logic system has been accompanied with the urge to *eliminate* it, where elimination consists in showing that all sequents derivable with cut can also be derivable without it. Cut elimination has since been linked with several logical and computational properties of a logic system; a well-known example is the Curry-Howard isomorphism [CF58, How80], which led to the Intuitionistic Type Theory of Martin Löf [ML79], in which cut elimination is shown to correspond to a β-elimination form of computation in typed λ-calculus.

doi:10.1093/jigpal/jzm040

It has been noted that the real importance of cut-free proofs is not the absence of cuts per see, but the fact that those proofs possess the subformula property [Smu68a]. Based on such observation, a few authors have started to explore the proof theory of *analytic cuts*, which allows cuts but limit their application so that the resulting proofs still have the subformula property [DM94]. The class of proofs thus obtained had interesting properties, such as analycity, the subformula property, and the irredundancy of proofs. However, the computational problem remained, namely, there exists some theorems with polynomially-sized proofs with general cuts, but for which only exponentially large analytic proofs are possible.

The urge for eliminating cut has stayed in the logic community. Cut elimination was introduced as a tool to prove the consistency of the sequent calculus, but only very few newly proposed logics have its consistency proved this way. The preferred road to proofs of consistency today is to provide a sound and complete semantics, which accounts for consistency. Nevertheless, the existence of a cut-free formulation for logics such as modal **S5** remains a much studied open problem since its proposal [Fit83], despite the fact that there are elegant and efficient proof systems for **S5**, e.g. [Mas00].

However, even if cut *can* be eliminated, this is only a property of the logic system, from which one should not infer that it *must* be eliminated. The role of the cut rule is absolutely central in foundational issues, for it corresponds to the use of auxiliary lemmas in the construction of large proofs of theorems, and Mathematics as we know it would be unimaginable without it.

All the power and elegance coming from cut elimination has a price. Cut free proofs tend to be redundant, with identical subtrees occurring in several parts of the proof. In classical propositional logic, there are several theorems for which small proofs can be easily obtained with cuts, but only exponentially large cut-free proofs exist [Boo84, CS00].

But there remains a *myth, which states that computing non-analytical cuts is very hard*. The intuition behind the statement against non-analytics cuts is that if they are allowed, the search space for cut formulas is in principle infinite. So, even though there may be shorter proofs in such a proof system, it may be much harder to find them.

The aim of this paper is to show that this is a false myth.

To motivate our position, we note that modern propositional theorem provers, which are generally called SAT-solvers, are becoming faster, with impressive practical results [MSS99, BM96, MMZ+01, GN02]. The majority of these SAT-solvers implement an improved version of the DPLL method (also called the Davis-Putnam method [DLL62]). It is known that the DPLL method can be seen as a clausal calculus based on a very limited version of analytic cut [D'A90], where the cut formula is always a literal. To obtain orders of magnitude of efficiency improvement, modern SAT solvers extend the basic DPLL with special heuristics, very smart data types, but also with techniques containing considerable proof insight, such as *conflict-driven learning* [ZMMM01] and *backjumping* [Dec90, NOT05]. It is our belief that those techniques used by SAT solvers are not simple, ad hoc hacks to obtain performance gains, but that they can be analysed and generalised in proof-theoretical terms.

In this paper, we wish to study proofs containing efficiently computable non-analytic cuts. For that, we show that any propositional classical sequent proof can be converted into a proof in a fixed format, the *comb-tree proof*. The comb-tree proof has exactly the same number of branches as the original proof, and each branching point is a cut that is potentially non-analytic. To show this result we start from a particular formulation of the sequent calculus in which *the cut rule is not eliminable*; we call it the *cut-based sequent calculus*. The presentation here is restricted to propositional classical logic, but can certainly

be extended to its extensions, such as first-order or modal and temporal logic, as well as to non-classical logic. In the cut-based calculus, the only branching rule is the cut so its use controls the structure of the proof.

It must be noted that to compute a sequent proof, one actually starts from the root sequent and proceeds towards the leaves, which is normally done using a tableau method. The results concerning non-analytic cuts in this paper are formulated in terms of tableaux and tableau methods. The tableau system corresponding to the cut-based calculus is the KE-tableau [DM94].

We show that there are other ways of introducing computable non-analytics cuts in a proof, which can be computed as one goes along building the proof. We show that this can be obtained by *learning* formulas when conflicts (tableau branch closures) are detected. The resulting proof may be a lot smaller that an analytic of cut-free proof, and is guaranteed to be polynomially bounded in the size of the initial proof.

KE-tableaux are the basis for our study of two forms of conflict-driven learning of formulas, namely *decision-based learning* and *inference-based learning*. We will show how these methods relate to the presence of non-analytic cuts in a proof and why the latter is a method more likely to reduce the size of a proof.

The rest of the paper develops as follows. Section 2 presents the cut-based calculus and its relation to KE-tableaux. The comb-tree normal form for KE-proofs is proven in Section 3. Conflict-driven learning and its relation to non-analytic cuts is explored in Section 4. We finalise with some conclusions and pointer to further work in Section 5.

2 Preliminaries

We deal here with a propositional language, $\mathcal{L}_P$, constructed from a countable set of atomic symbols $\mathcal{P}=\{p_0,p_1,\ldots\}$, and the usual classical connectives $\neg$, $\wedge$, $\vee$ and $\rightarrow$. The size of a formula A, $\|A\|$, is the total number of symbol occurrences in A, defined inductively as $\|p_i\|=1$, $\|\neg A\|=1+\|A\|$ and $\|A\circ B\|=1+\|A\|+\|B\|$ for $\circ\in\{\wedge,\vee,\rightarrow\}$. A *valuation* is a function $v:\mathcal{P}\rightarrow\{0,1\}$ that is extended to all formulas in the usual way: $v:\mathcal{L}_P\rightarrow\{0,1\}$. A formula A is *satisfied* by v if $v(A)=1$; A is *satisfiable* if there exists a v that satisfies A; A is *valid* if it is satisfied by all valuations v. Let Γ and Δ be sets of formulas, then we write $\Gamma\models\Delta$ if every valuation that satisfies every $A\in\Gamma$ satisfies some $B\in\Delta$.

2.1 A Cut-based System for Classical Logic

We present here a version of the sequent calculus in which cut is not eliminable, inspired by the ideas of D'Agostino and Mondadori [D'A92, DM94], which has been explored in [FG06]. Our presentation follows the discipline of Kleene's **G4** sequent system for classical logic [Kle67], such that a sequent is a pair $\Gamma\vdash\Delta$, where Γ and Δ are *sets* of formulas, with the intended meaning that the conjunction of the formulas in Γ prove the disjunction of the formulas in Δ. The *structural rules*, i.e. the rules that manipulate sequents independently of the connectives in their formulas, are presented in Figure 1. Note that since both the antecedent and consequent of a sequent are sets, there is no need for the usual rules of commutativity and contraction. Also, the usual rule of monotonicity or weakening is taken care by the Axiom rule. The cut rule is crucial here, as it cannot be eliminable. In the Axiom rule, we call the distinguished formula A the *main* formula and any formula in Γ and Δ is a *weak* formula; in the cut rule, the distinguished formula A is called the *cut formula*.

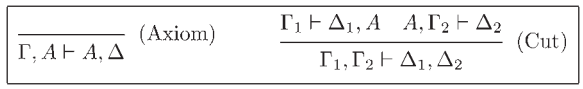


FIG. 1. Classical structural rules for the sequent calculus

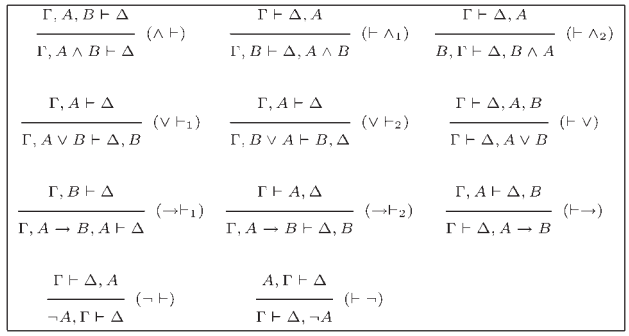


FIG. 2. Connective rules for the cut-based sequent calculus

The connective rules of the sequent calculus are different from the usual ones, so as to block cut elimination. Those connective rules are presented in Figure 2. Instead of being a cut-free calculus, this is a *cut-based* calculus.

A *deduction* of *proof* in the cut-based sequent calculus is a tree whose nodes are sequents, where the leaf nodes are instances of the Axiom rule, and the internal nodes are derived from other nodes using the inference rules in Figures 1 and 2. A sequent is *deducible* or *derivable* if it is the root of a deduction tree; if $\Gamma \vdash \Delta$ is deducible in the cut-based sequent calculus, we write $\Gamma \vdash_{\text{CB}} \Delta$.

Note that a node has two parent nodes only if it is a result of applying the Cut Rule. Otherwise the node is a *linear successor* of it single parent. A *supernode* is a maximal sequence of linear successors. It is easy to see that a deduction tree can be seen as binary tree of supernodes.

All sequent rules in Figures 1 and 2, like those of Kleene's **G4** system, are *invertible*, that is, the provability of a rule's conclusion implies the provability of all the rule's premisses. This important property can be easily verified in semantic terms.

PROPOSITION 2.1 (Soundness and Completeness of the Cut-based Calculus)
$\Gamma \vdash_{\text{CB}} \Delta$ iff $\Gamma \models \Delta$.

Soundness can be proved directly, by showing that axioms are valid and if the inference in rules in Figures 1 and 2 are applied to valid sequents the derived sequent is always valid. Completeness can be shown in many ways; in particular, it was shown in [FG06] that the rules in the cut-based calculus can simulate those of a traditional sequent calculus and thus inherits its completeness.

We next show that cut cannot be eliminated in the cut-based systems.

LEMMA 2.2
There are classically valid sequents which are not deducible without the use of the cut rule in the cut-based calculus.

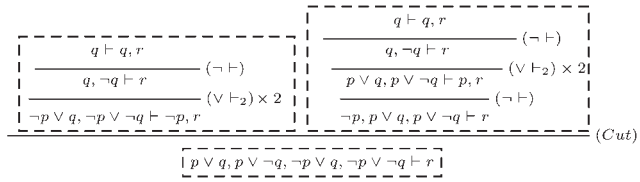


FIG. 3. A Cut-based Deduction

PROOF. Consider the classically valid sequent $p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q \models r$. By completeness, we know that $p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q \vdash_{\text{CB}} r$ (*). We argue that the last rule applied in the deduction of such theorem has to be the cut rule.

In fact, (*) is not an axiom. There are no connectives in its consequent and all formulas in its antecedent have $\vee$ as the main connective. The only connective rules that introduce an $\vee$ in the antecedent are $(\vee\vdash_1)$ and $(\vee\vdash_2)$, so the last rule applied in the sequents deduction must have been either $(\vee\vdash_1)$ or $(\vee\vdash_2)$ or cut. However, rules $(\vee\vdash_1)$ and $(\vee\vdash_2)$ also add one of the disjuncts to the consequent. As the consequent of (*) is an atom not occurring in its antecedent, $(\vee\vdash_1)$ and $(\vee\vdash_2)$ cannot have been applied. So cut was applied last. ∎

A cut-based deduction of sequent (*) is shown in Figure 3. The last step is a cut over $\neg p$. The dashed boxes indicate the supernodes of the proof.

Given a cut-based proof Π of a sequent $\Gamma \vdash_{\text{CB}} \Delta$, we define the *size* of Π, $|\Pi|$, as the number of sequents in Π. Let $c(\Pi)$ be the number of cuts in Π; then the number of supernodes in Π is $2c(\Pi)+1$, and the number of branches is $c(\Pi)+1$. For example, with regards to the sequent proof Π_0 in Figure 3, we have that $|\Pi_0|=7$, the number of cuts $c(\Pi_0)=1$, the number of supernodes is 3 and the number of branches is 2.

2.2 KE Tableaux

KE-tableaux correspond directly to cut-based systems. This form of tableau was proposed by D'Agostino and Mondadori as a way of incorporating directly the cut principle in tableau proofs, which they argue correspond to the principle of excluded middle, also called the *principle of bivalence* (PB) [DM94, D'A99]. Furthermore, it was shown that, in comparison with Smullyan's analytic tableau based on cut-free sequents [Smu68b], KE-tableau have better computational properties [D'A92]. We note, however, that D'Agostino's presentation of KE-tableaux was based on semantical considerations of non-redundancy over Kleene's **G4** sequent system; here, we show that the same tableau system can be obtained simply by inverting the sequent rules of the cut-based calculus.

KE-tableaux deal with T- and F-signed formulas. So if α is a formula, $T\ \alpha$ and $F\ \alpha$ are signed formulas. $T\ \alpha$ is the *conjugate formula* of $F\ \alpha$, and vice versa; consider the *conjugate operator* that applies to formula sings such that $\bar{T} = F$ and $\bar{F} = T$. Each connective is associated with a set of *linear expansion rules*. Linear expansion rules always have a *main premiss*; two-premissed rules also have an *auxiliary premiss*. Figure 4 shows KE-tableau linear connective expansion rules for classical logic. It is no coincidence that there is a one-to-one correspondence with the cut-based sequent rules in Figure 2.

The only branching rule in KE is the *Principle of Bivalence* (PB), stating that a formula A must be either true or false, as illustrated in Figure 5, which corresponds directly to the

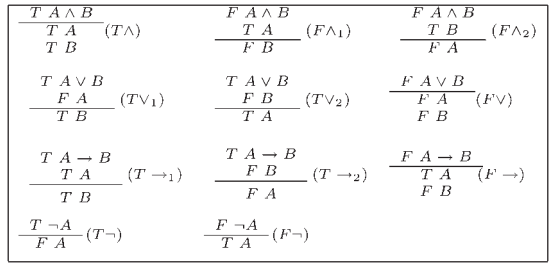


FIG. 4. KE Expansion Rules

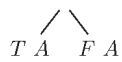


FIG. 5. Principle of Bivalence

cut rule. D'Agostino [D'A92] restricted the use of PB to the insertion of auxiliary premisses of two-premissed rules; This restriction was called the *branching heuristics* and corresponds to allow only analytic cuts in a proof. Analycity limits the potentially infinite search space for a cut formula, without loss of completeness.

A tableau for sequent $A_1,\ldots,A_n \vdash B_1,\ldots,B_m$ starts with a single branch containing $TA_1,\ldots,TA_n,FB_1,\ldots,FB_m$. An expansion of a tableau branch is allowed when the premisses of an expansion rule are present in the branch; the expansion consists of adding the conclusions of the rule to the end of all branches passing through the set of all premisses of that rule. The PB branching rule splits a branch into two.

A branch is *closed* if it contains $F\ A$ and $T\ A$ for some A. The tableau is closed if all its branches are closed. The inference $\vdash_{\mathbf{KE}}$ is defined for $A_1,\ldots,A_n \vdash_{\mathbf{KE}} B_1,\ldots,B_m$ iff there is a closed KE-tableau $\mathcal{T}$ for $TA_1,\ldots,TA_n,FB_1,\ldots,FB_m$.

A KE tableau proof is therefore a tree, whose nodes are signed formulas. There are two measures of complexity of a tableau. The *size* of a tableau $\mathcal{T}$, $|\mathcal{T}|$, is the number of nodes in $\mathcal{T}$, and $c(\mathcal{T})$ is the number of applications of PB in $\mathcal{T}$. The *number of symbols* of $\mathcal{T}$, $\|\mathcal{T}\|$, is the sum of the sizes of all formulas in all nodes of $\mathcal{T}$: $\|\mathcal{T}\| = \sum_{XA\in\mathcal{T}, X\in\{T,F\}} \|A\|$. We define a *supernode* as a maximal sequence of nodes resulting from applications of linear expansion rules only.

We call a sequent proof *cut-free* if it employs no cuts; it is *analytic* if all cuts are analytic, that is, if the cut formula is a subformula of some formula in the sequent; it is *non-analytic* if there is no restriction on cuts. A similar terminology is used for tableaux.

We establish the correspondence between cut-based sequent deductions and KE tableaux.

LEMMA 2.3
For every cut-based proof Π for $A_1,\ldots,A_n \vdash_{\mathbf{CB}} B_1,\ldots,B_m$ there corresponds a tableau $\mathcal{T}$ for $A_1,\ldots,A_n \vdash_{\mathbf{KE}} B_1,\ldots,B_m$.

PROOF. Let S_n be a sequent in Π such that $\mathcal{S}_n = S_1 \cdots S_n$ is the sequence of sequents in Π from the root sequent S_1 to S_n. Define $\mathcal{B}_n = \{TA|\Gamma, A \vdash \Delta \in \mathcal{S}_n\} \cup \{FB|\Gamma \vdash B, \Delta \in \mathcal{S}_n\}$. We prove by induction that $\mathcal{B}$ is a well formed partially expanded KE tableau branch. In fact, this is indeed the case if we consider the root tableau, for then $\mathcal{B} = \{TA_1,\ldots,TA_n,FB_1,\ldots,FB_m\}$.

TABLE 1. Sequent-Tableau Correspondence

Correspondence Between Sequent Rules and Tableau Expansion Rules			
$(\wedge\vdash)$—$(T\wedge)$	$(\vee\vdash_1)$—$(T\vee_1)$	$(\rightarrow\vdash_1)$—$(T\rightarrow_1)$	$(\neg\vdash)$ —$(T\neg)$
$(\vdash\wedge_1)$—$(F\wedge_1)$	$(\vee\vdash_2)$—$(T\vee_2)$	$(\rightarrow\vdash_2)$—$(T\rightarrow_2)$	$(\vdash\neg)$ —$(F\neg)$
$(\vdash\wedge_2)$—$(F\wedge_2)$	$(\vdash\vee)$—$(F\vee)$	$(\vdash\rightarrow)$—$(F\rightarrow)$	

If S_n was obtained from S_{n+1} by a linear inference rule R, let E be the tableau rule corresponding to R in Table 1. Consider the sequence $\mathcal{S}_{n+1} = S_1 \cdots S_n S_{n+1}$; it is easy to note that $\mathcal{B}_{n+1}$ is obtained from $\mathcal{B}_n$ by an application of E, so $\mathcal{B}_{n+1}$ is a well formed partially expanded branch.

If $S_n = \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2$ was obtained by a cut from $S_{n_1} = \Gamma_1 \vdash \Delta_1, A$ and $S_{n_2} = A, \Gamma_2 \vdash \Delta_2$, then we construct the corresponding $\mathcal{B}_{n_1} = \mathcal{B}_n \cup \{FA\}$ and $\mathcal{B}_{n_2} = \mathcal{B}_n \cup \{TA\}$ as the two branches obtained from $\mathcal{B}_n$ by an application of PB, so both $\mathcal{B}_{n_1}$ and $\mathcal{B}_{n_2}$ are well formed partially expanded branches, which finishes the induction.

As all branches in Π end in an axiom node $S_n = \Gamma, A \vdash A, \Delta$, a fully extended branch $\mathcal{B}_n$ contains both FA and TA, so every branch closes. Two branches that have a common prefix can be *merged*, thus forming a binary tree; a branch can be merged into a tree by merging it to one of its branches, generating a larger tree. By the construction above, two branches of Π that are separated by a cut correspond to two fully expanded tableau branches that share a common prefix. The tableau $\mathcal{T}$ is then the merge of all fully expanded branches. ∎

The converse of Lemma 2.3 also holds.

LEMMA 2.4
For every KE tableau proof $\mathcal{T}$ for $A_1, \ldots, A_n \vdash_{\mathbf{KE}} B_1, \ldots, B_m$ there corresponds a cut-based sequent proof Π for $A_1, \ldots, A_n \vdash_{\mathbf{CB}} B_1, \ldots, B_m$.

PROOF. By induction on the construction of $\mathcal{T}$. The idea is to associate, for every partially expanded branch $\mathcal{B}$, the sequent $\{A \mid TA \in \mathcal{B}\} \vdash \{B \mid FB \in \mathcal{B}\}$. In this way, the initial sequent corresponds to the initial tableau. Suppose the tableau has been expanded up to a certain point in a branch $\mathcal{B}$, and the sequent proof has been constructed up to sequent $\Gamma \vdash \Delta$.

In case a linear expansion rule E is applied to $\mathcal{B}$ generating $\mathcal{B}'$, we form a new sequent $\Gamma' \vdash \Delta'$ based on $\mathcal{B}'$; clearly $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$. Let R be the sequent rule corresponding to E in Table 1. It is easy to verify that if E is applied to $\mathcal{B}'$ generating $\mathcal{B}'$, then R applied to $\Delta \subseteq \Delta'$ generates $\Gamma \vdash \Delta$, so the sequent proof was expanded in a sound way.

In case the PB branching rule is applied to $\mathcal{B}$ on formula A, generating $\mathcal{B}_1 = \mathcal{B} \cup \{FA\}$ and $\mathcal{B}_2 = \mathcal{B} \cup \{TA\}$, a cut rule is inserted in the sequent proof with premisses $\Gamma \vdash \Delta, A$ corresponding to $\mathcal{B}_1$ and $A, \Gamma \vdash \Delta$ corresponding to $\mathcal{B}_2$. As the cut generates $\Gamma \vdash \Delta$, the sequent proof is expanded in a sound way.

Finally, if $\mathcal{B}$ is closed, then there are $FA, TA \in \mathcal{B}$, so the correspondent sequent is an axiom of the form $\Gamma'', A \vdash A, \Delta''$, and the sequent proof has a sound branch. As all branches in $\mathcal{T}$ close, we have a correct sequent proof for the initial sequent. ∎

One important aspect of the transformations of Lemmas 2.3 and 2.4 is that they preserve the shape of the proof, that is, the number of branches, the branching points and the number of rule applications in each branch is preserved. Suppose $\triangleright$ represents the transformation of

Lemma 2.3, and ▶ represents the transformation of Lemma 2.4. Note that if we start with a KE tableau $\mathcal{T}$, then

$$\mathcal{T} \triangleright \Pi \blacktriangleright \mathcal{T},$$

that is, if one applies the transformation in Lemma 2.4 and then applies the transformation of Lemma 2.3, one ends up with the same original tableau $\mathcal{T}$; however, if one starts with a sequent proof Π, and apply the transformations

$$\Pi \blacktriangleright \mathcal{T} \triangleright \Pi',$$

one ends up with a sequent proof Π' possibly different from Π. However, Π and Π' are very similar, due to the following:

- The root nodes of Π and Π' are identical; that is Π and Π' prove the same sequent.
- Every leaf node $\Gamma, A \vdash A, \Delta$ in Π corresponds to a sequent node $\Gamma' \vdash \Delta'$ in Π', such that $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$; this is due to the fact that both such nodes correspond to the closing of a tableau branch with TA and FA, but the $\triangleright$-transformation may add extra formulas to the axiom sequents.
- If sequent S was obtained from $S_1, \ldots S_n$ in Π with rule R, then in Π' this corresponds to a node S' obtained from $S'_1, \ldots S'_n$ with the same rule R.
- As a consequence, every sequent node $\Gamma \vdash \Delta$ in Π corresponds to a sequent node $\Gamma' \vdash \Delta'$ in Π', such that $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$.

Corollary 2.5
$\vdash_{\mathbf{KE}}$-inference is sound and complete w.r.t. classical logic.

As an example, Figure 6 presents a KE tableau for $p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q \vdash r$, corresponding to the sequent deduction in Figure 3. The dashed boxes indicate the supernodes of the proof.

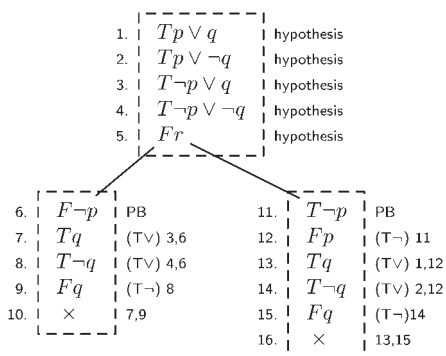


Fig. 6. A KE-Tableau Deduction

3 Normal Form for Non-Analytic Proofs

As there is a direct correspondence between KE-tableaux and cut-based sequent proofs, we restrict our attention only to KE-tableaux.

A deduction is a binary tree, but trees can degenerated in linear-like structures which we call *combs*. A right-branching comb is a binary tree in which, at every branching point the left node is always a leaf; similarly, in a left-branching tree, at every branching point the right node is always a leaf.

3.1 A Normal Form for KE-Tableaux

We say that $\mathcal{T}$ is a *comb tableau* if the supernodes of $\mathcal{T}$ form a comb tree. As KE trees correspond to sequent proofs turned upside-down, a left-branching comb tree sequent proof corresponds to a right-branching comb tree KE tableau, and vice-versa.

We start with an example of how an analytic tableau proof, can be transformed into a comb tableau. Consider the valid sequent $p\vee q, q\rightarrow p, (p\wedge t)\rightarrow s, p\rightarrow (t\vee s)\vdash \neg(\neg p\vee\neg s)$. A first, analytic KE-tableau for it is shown in Figure 7.

A comb tree KE-tableau proof for the same sequent is shown in Figure 8. Node 22 was obtained with rule $(T\wedge\rightarrow)$, described below. We stress that the proofs correspond to each other; in fact, the proof in Figure 8 was obtained from the proof in Figure 7 by means of Algorithm 3.1, to be presented below.

THEOREM 3.1
For every closed tableau $\mathcal{T}$ for $\Gamma\vdash_{\mathbf{KE}}\Delta$ there corresponds a comb tree closed tableau $\mathcal{T}'$ for $\Gamma\vdash_{\mathbf{KE}}\Delta$, such that $c(\mathcal{T}')=c(\mathcal{T})$ $|\mathcal{T}'|=O(|\mathcal{T}|^2)$ and $\|\mathcal{T}'\|=O(\|\mathcal{T}\|^3)$. Furthermore, there is an algorithm for constructing $\mathcal{T}'$ from $\mathcal{T}$.

PROOF. To obtain a one-to-one correspondence between the branches of $\mathcal{T}$ and $\mathcal{T}'$, we will assume as primitive the following rules $(T\wedge\rightarrow)$ and $(T\rightarrow\vee)$, which are generalisations,

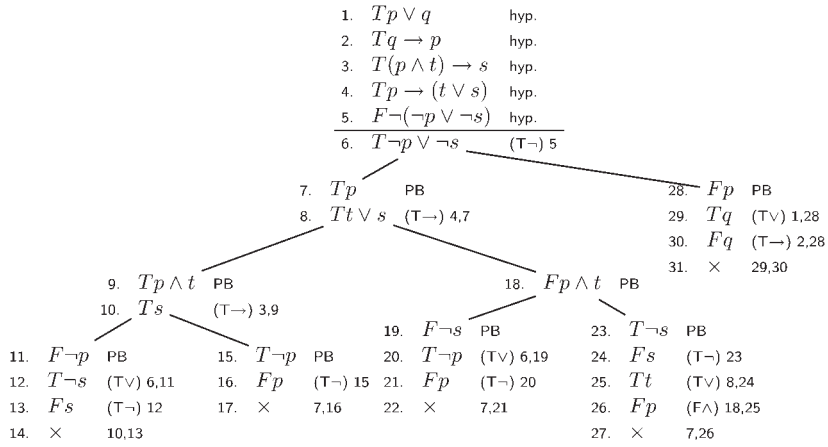


FIG. 7. A Tableau Proof

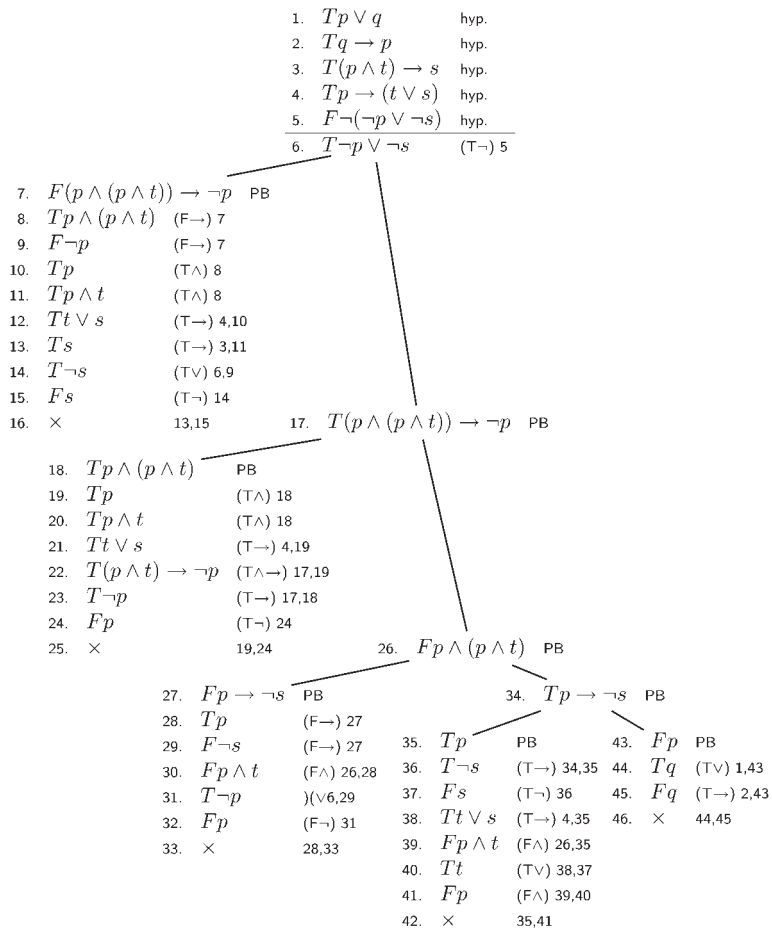


FIG. 8. A Corresponding Comb Tableau

respectively, of $(T{\to}_1)$ and $(T{\to}_2)$:

$$\frac{\begin{array}{l} T(A_1 \wedge \ldots \wedge A_\ell) \to B \\ TA_i \end{array}}{T(A_1 \wedge \ldots \wedge A_{i-1} \wedge A_{i+1} \wedge \ldots \wedge A_\ell) \to B}\,(T\wedge\to)$$

$$\frac{\begin{array}{l} TA_1 \to (B_1 \vee \ldots \vee B_k) \\ FB_j \end{array}}{TA \to (B_1 \vee \ldots \vee B_{j-1} \vee B_{j+1} \vee \ldots \vee B_k)}\,(T\to\vee)$$

Rules $(T\wedge\to)$ and $(T\to\vee)$ can be easily derived using a single application of PB, which guarantees that the comb-tree structure will not be affected even if they are not assumed to be primitive. In fact, the formula X:

$$X = ((A_1 \wedge \ldots \wedge A_\ell) \to B) \to (A_i \to ((A_1 \wedge \ldots \wedge A_{i-1} \wedge A_{i+1} \wedge \ldots \wedge A_\ell) \to B))$$

is a classical theorem that can be proved first thing in a tableau proof $\mathcal{T}''$, such that if $\mathcal{T}'$ is comb-like, the resulting tableau $\mathcal{T}''$ remains comb-like:

$\mathcal{T}'' =$

		PB		
n_1	FX		n_1	TX PB
n_2	$T(A_1 \wedge \ldots \wedge A_\ell) \rightarrow B)$	$(\mathsf{F}\rightarrow)\ n_1$		$\mathcal{T}'$
n_3	TA_i	$(\mathsf{F}\rightarrow)\ n_1$		
n_4	$TA_1 \wedge \ldots \wedge A_{i-1} \wedge A_{i+1} \wedge \ldots \wedge A_\ell$	$(\mathsf{F}\rightarrow)\ n_1$		
n_5	FB	$(\mathsf{F}\rightarrow)\ n_1$		
n_6	TA_1	$(\mathsf{T}\wedge)\ n_4$		
$\vdots$	$\vdots$	$\vdots$		
$n_{6+\ell-1}$	TA_ℓ	$(\mathsf{T}\wedge)\ n_4$		
$n_{6+\ell}$	$FA_1 \wedge \ldots \wedge A_\ell$	$(\mathsf{F}\rightarrow)\ n_2, n_5$		
$n_{6+\ell+1}$	FA_ℓ	$(\mathsf{F}\wedge)\ n_{6+\ell}, n_3, n_6, \ldots, n_{6+\ell-2}$		
	$\times$	$n_{6+\ell-1}, n_{6+\ell+1}$		

Note that in the right branch, due to the presence of TX, if the premisses of $(T\wedge\rightarrow)$ are in the branch, so is its conclusion. An analogous construction can be done with respect to $(T\rightarrow\vee)$, so that the proof $\mathcal{T}'$ can be prefixed as many instances of the construction above as are the instances of $(T\wedge\rightarrow)$ and $(T\rightarrow\vee)$ in $\mathcal{T}'$ such that, if $\mathcal{T}'$ comb-like so is the resulting tableau, as desired.

Let $\mathcal{T}$ be a closed tableau proof tree; an order is imposed on the sub-trees of $\mathcal{T}$ such that at every branching point, the left subtree is ordered *before* the right subtree. This ordering is supposed to encode the fact that tableau trees are usually constructed as a depth-first search, where the application of PB-branching over A is usually the result of the application of some heuristic method, which chooses both the formula A over which the branching will occur and the polarity that will be explored first, that is, if TA or FA will be the "first to be explored". We call the top node of the first sub-tree to be explored as the *left choice* formula of that PB application, and the top node of the other sub-tree is the *right choice* formula. Given a branch $\mathcal{B}$ in $\mathcal{T}$, let $\texttt{lChoice}(\mathcal{B})$ and $\texttt{rChoice}(\mathcal{B})$ represent, respectively, the set of all left choice signed formulas in $\mathcal{B}$, and the set of all right choice formulas in $\mathcal{B}$. Also, let $\texttt{Choice}(\mathcal{B}) = \texttt{lChoice}(\mathcal{B}) \cup \texttt{rChoice}(\mathcal{B})$.

The order on subtrees also imposes an order on the branches of $\mathcal{T}$, $\mathcal{B}_1, \ldots, \mathcal{B}_b$, such that $\mathcal{B}_i$ precedes $\mathcal{B}_j$ if at some branching point in $\mathcal{T}$, $\mathcal{B}_i$ contains the left choice formula and $\mathcal{B}_j$ contains the corresponding right choice formula.

Suppose $\mathcal{T}$ is a closed tableau for $A_1, \ldots, A_n \vdash_{\mathbf{KE}} B_1, \ldots, B_m$. All branches in $\mathcal{T}$ extend the initial branch $\mathcal{B}_0 = \{TA_1, \ldots, TA_n, FB_1, \ldots, FB_m\}$, and every branch $\mathcal{B}_i$ in $\mathcal{T}$ can be reconstructed from $\mathcal{B}_0$ and $\texttt{Choice}(\mathcal{B}_i)$. In fact, we can expand $\mathcal{B}_0 \cup \texttt{Choice}(\mathcal{B}_i)$ applying all possible linear expansion rules; let us call the resulting set $\texttt{LinearExpand}(\mathcal{B}_0 \cup \texttt{Choice}(\mathcal{B}_i))$. Clearly, $\mathcal{B}_i \subseteq \texttt{LinearExpand}(\mathcal{B}_0 \cup \texttt{Choice}(\mathcal{B}_i))$, so if the former closes the latter will certainly close.

The main idea of the proof is that, through successive uses of PB a branch $\mathcal{B}$ can be obtained not from $\texttt{Choice}(\mathcal{B}_i)$, but from $\texttt{lChoice}(\mathcal{B}_i)$, i.e. only the left choice formulas in $\mathcal{B}$, which will yield a right-branching comb. A left-branching comb could be obtained by considering $\texttt{rChoice}(\mathcal{B}_i)$ in a similar way.

For each branch $\mathcal{B}_i$, let $\texttt{lChoice}(\mathcal{B}_i) = \Lambda_i^T \cup \Lambda_i^F$ be a partition of the set $\texttt{lChoice}(\mathcal{B}_i)$, where $\Lambda_i^T = \{A \mid TA \in \texttt{lChoice}(\mathcal{B}_i)\}$ and $\Lambda_i^F = \{A \mid FA \in \texttt{lChoice}(\mathcal{B}_i)\}$. Define the formula,

for $1 \leq i \leq b-1$:

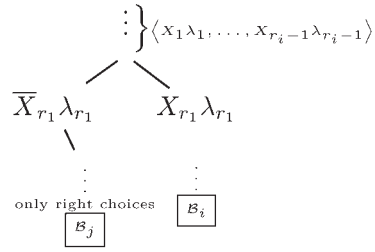
$$\lambda_i^{PB} = \begin{cases} \bigwedge \Lambda_i^T, & \text{if } \Lambda_i^F = \varnothing \\ \bigvee \Lambda_i^F, & \text{if } \Lambda_i^T = \varnothing \\ \left(\bigwedge \Lambda_i^T\right) \rightarrow \left(\bigvee \Lambda_i^F\right) & \text{otherwise} \end{cases}$$

As we cannot have simultaneously $\Lambda_i^T = \varnothing = \Lambda_i^F$, λ_i^{PB} is a well-formed propositional formula; as $\Lambda_b^T = \varnothing = \Lambda_b^F$, λ_b^{PB} is not defined. Consider the *left polarity*, L_p, of λ_i^{PB} to be $L_p = F$ if $\Lambda_i^F \neq \varnothing$, and $L_p = T$ otherwise. The idea is that, if we consider the signed formula $L_p \lambda_i^{PB}$, $\texttt{LinearExpand}(\{L_p \lambda_i^{PB}\}) \supseteq \texttt{lChoice}(\mathcal{B}_i)$. We are now in a position to prove the correctness of Algorithm 3.1.

Algorithm 3.1 shows how to construct a closed, comb-tree tableau $\mathcal{T}'$ starting from a closed $\mathcal{T}$. For every branch $\mathcal{B}_i$ in $\mathcal{T}$ there corresponds a branch $\mathcal{B}'_i$ in $\mathcal{T}'$; all those branches extend $\mathcal{B}_0$. Each $\mathcal{B}'_i$, $1 \leq i \leq b-1$ contains a single left polarity formula, namely $L_p\ \lambda_i^{PB}$, so $\mathcal{T}'$ is a comb tree. We now prove by induction that $\mathcal{B}'_i \supseteq \texttt{Choice}(\mathcal{B}_i)$, which entails that $\mathcal{B}'_i$ is a closed branch.

For the base case, consider $\mathcal{B}'_1$ which, according to line 3 in Algorithm 3.1, contains $\{L_p\ \lambda_1^{PB}\}$, so that its linear expansion will contain all left choice formulas in $\mathcal{B}_1$, which contains no right choice formulas, so the result is proved.

Now suppose $\mathcal{B}'_j \supseteq \texttt{Choice}(\mathcal{B}_j)$ for $1 \leq j < i$ and prove that the same holds for $\mathcal{B}'_i$. We first note that the linear expansion of $L_p \lambda_i^{PB}$ contains all left choice formulas in $\mathcal{B}_i$ and, from line 5 in Algorithm 3.1, so does $\mathcal{B}'_i$. So we need to show that $\mathcal{B}'_i$ also contains the right choice formulas in $\mathcal{B}_i$. Let $\texttt{Choice}(\mathcal{B}_i) = \langle X_1 \lambda_1, \ldots, X_{\beta_i} \lambda_{\beta_i} \rangle$ be the ordered sequence of $\mathcal{B}_i$'s signed choice formulas, $X_k \in \{T, F\}, 1 \leq k \leq \beta_i$. Let $X_{r_1} \lambda_{r_1}$ be the first right choice in that sequence. Then in $\mathcal{T}$ we have the following configuration:



We see that at some point in $\mathcal{T}$, there is a PB-branching over λ_{r_1}, such that $\mathcal{B}_i$ extends the branch containing $X_{r_1} \lambda_{r_1}$. On the left subtree, there is guaranteed to exist a branch $\mathcal{B}_j$ where all subsequent branching choices, if any, are right choices; in fact, this is true of all left branches in a KE-tableau proof. Suppose $X_{r_1} = F$. Then, on branch $\mathcal{B}'_j$ in $\mathcal{T}'$ the left branch choice is $F(\lambda_{r_1} \wedge A_1 \wedge \ldots \wedge A_\ell) \rightarrow (B_1 \vee \ldots \vee B_k)$, where $A_1, \ldots, A_\ell$ and $B_1, \ldots, B_k$ are a regrouping of $\lambda_1, \ldots, \lambda_{r_1 - 1}$ according to the polarity of X_a, $1 \leq a < r_1$. Therefore $\mathcal{B}'_i$ contains $T(\lambda_{r_1} \wedge A_1 \wedge \ldots \wedge A_\ell) \rightarrow (B_1 \vee \ldots \vee B_k)$. As we know that the linear expansion of λ_i^{PB} contains $X_1 \lambda_1, \ldots, X_{r_1 - 1} \lambda_{r_1 - 1}$, by $\ell + k$ applications of the rules $(T \wedge \rightarrow)$, $(T \rightarrow \vee)$, $(T \rightarrow_1)$ and $(T \rightarrow_2)$ leads to $F \lambda_{r_1} \in \mathcal{B}'_i$. A totally analogous argument shows that if $X_{r_1} = T$, then $F \lambda_{r_1} \in \mathcal{B}'_i$.

We have thus shown that the first right branching point of $\mathcal{B}_i$ is in $\mathcal{B}'_i$. Similarly, we can prove now that if the $(r-1)$ first right branching points of $\mathcal{B}_i$ are in $\mathcal{B}'_i$, so is the r-th right branching point. The argument is totally analogous to the above, so we omit the details.

This finishes the induction, and we have thus proved that $\mathcal{B}'_i \supseteq \texttt{Choice}(\mathcal{B}_i)$, so for $1 \leq i \leq b-1$, $\mathcal{B}'_i$ closes and the comb-tree tableau $\mathcal{T}'$ computed by algorithm 3.1 closes; it follows that Algorithm 3.1 is correct.

We now see that for every application of PB in $\mathcal{T}$ there corresponds exactly one application of PB in $\mathcal{T}'$, so $c(\mathcal{T}') = c(\mathcal{T})$. Note that, if we had not used rules $(T\wedge\rightarrow)$ and $(T\rightarrow\vee)$ as primitive, but had to derive them, we would need one such a derivation for each branching point in $\mathcal{T}'$, which would give us $c(\mathcal{T}') = O(c(\mathcal{T}))$.

With regards to the number of nodes of $\mathcal{T}'$, we see that each $\mathcal{B}'_i$ is basically $\mathcal{B}_i$, enlarged with the presence of $\{L_p\ \lambda_i^{PB}\} \cup \{\overline{L_p}\ \lambda_j^{PB} | 1 \leq j < i\}$, according to line 5 in Algorithm 3.1. The computation of the choice formulas in $\mathcal{B}'_i$ takes $\sum_{1\leq j\leq i} |\lambda_j^{PB}|$ and since $|\lambda_j^{PB}| = O(|\mathcal{B}_j|)$, it follows that $|\mathcal{T}'| = O(|\mathcal{T}|^2)$. Finally, for the number of symbols in $\mathcal{T}'$, it suffices to note that, as $|\lambda_j^{PB}| = O(|\mathcal{B}_j|)$, the number of symbols in a branch $\mathcal{B}'_j$ is at most $|\mathcal{B}'_j|^2$, so it follows that $\|\mathcal{T}'\| = O(\|\mathcal{T}\|^3)$. ∎

Algorithm 3.1 NormalFormTableau($\mathcal{T}$, $\Gamma \vdash_{\textbf{KE}} \Delta$)

Input: $\mathcal{T}$, a closed tableau proof for $\Gamma \vdash_{\textbf{KE}} \Delta$.
Output: $\mathcal{T}'$, a closed, right-branching comb-tree tableau proof for $\Gamma \vdash_{\textbf{KE}} \Delta$.

1: Let $\mathcal{B}_1, \ldots, \mathcal{B}_b$ be the branches of $\mathcal{T}$, whose initial branch is $\mathcal{B}_0$;
2: We construct $\mathcal{T}'$ with branches $\mathcal{B}'_1, \ldots, \mathcal{B}'_b$ extending $\mathcal{B}_0$;
3: $\mathcal{B}'_1 = \texttt{LinearExpand}(\mathcal{B}_0 \cup \{L_p\ \lambda_1^{PB}\})$;
4: **for** $i = 2$ to $b-1$ **do**
5: $\quad \mathcal{B}'_i = \texttt{LinearExpand}(\mathcal{B}_0 \cup \{L_p\ \lambda_i^{PB}\} \cup \{\overline{L_p}\ \lambda_j^{PB} | 1 \leq j < i\})$;
6: **end for**
7: $\mathcal{B}'_b = \texttt{LinearExpand}(\{\overline{L_p}\ \lambda_i^{PB} | 1 \leq i \leq b-1\} \cup \mathcal{B}_0)$;
8: The tree $\mathcal{T}'$ is the result of the merge of $\mathcal{B}'_1, \ldots, \mathcal{B}'_b$.
9: **return** $\mathcal{T}'$;

4 Learning

D'Agostino has noted that KE systems can be viewed as a generalised form of Davis-Putnam procedure [D'A90]. The Davis-Putnam procedure [DLL62], despite being restricted to clausal logic, is widely used in many theorem provers, and has received several improvements over the years. It is our goal here to understand those improvements in a principled way and see how they can be applied to KE tableaux.

The Davis-Putnam procedure is usually associated with the construction of a SAT-solver, which aims at finding a valuation that satisfies a formula; a tableaux for SAT-solving would search for an open branch, considering closed branches as dead-ends. On the other hand, theorem proving searches for a proof of a formula, and if tableaux are viewed as theorem provers, closed branches are the sub-goals. There is, in fact, no contradiction between these two points of view, as tableaux (as well as the Davis-Putnam procedure, or resolution, among many others) and are decision procedures, whose design must aim at finding an open branch as soon as possible, if one exists; otherwise, the methods should find a small proof tree if all branches will eventually close. In particular, pure analytic tableaux fail for the latter goal, as

there are no small (i.e., polynomial) proofs for several theorems which are known to possess polynomial non-analytic proofs.

In automated theorem proving, several techniques have been applied to obtain smaller proof graphs and quicker proof searches. One of the techniques most largely employed is called *learning*. In the context of proof search, learning means the inference of new formulas during the search process, which are then added to the set of given formulas, and which can be later discarded.

Typically, learning takes places in refutation-style proof constructions, when a conflict is reached in a search path; in a tableau, a conflict is in fact the presence of a contradiction in a branch, which corresponds to the closing of this branch. One then computes some information, in the form of a new formula to be added to the root of the tableau, that will be used by the search engine to avoid reaching again the same conflict. We explore here two techniques used to that effect:

- Learning based on the (bad) decisions that led to the conflict; we call this *decision-based learning*.
- Learning is based on some form of inference involving the formulas that took part in the production of the conflict, typically resolving the two main formulas that produced the contradiction; we call this *inference-based learning*.

It must be clear that those learning techniques could be applied to analytic tableaux as well, simply as methods for adding new formulas every time a branch closes. The difference is that, unlike analytic tableaux, each learning method can be reconstructed (or simulated) as a KE-tableau proof.

We now explore these two learning techniques in the context of KE-tableaux.

4.1 Decision-based Learning

Most inference systems are non-deterministic, in the sense that at several points during the proof construction a decision has to be taken about which inference to choose. Normally, the correctness of the proof is not affected by this choice, but it can have a dramatical influence on the proof size.

In KE-tableaux, one tries to apply all possible linear expansion rules, and then, if the branch is not closed nor saturated, the only possibility is to apply the PB branching rule. But then comes the real choice, namely, which signed formula will be on the left branch. The left choice formula is the only choice formula at a branching point with a depth-first search strategy, for the right choice formula will only be explored if the left branch is closed, at which point no choice is left.

Decision-based learning adds the learned signed formula to the root of the tableau, as if it were a "new" hypothesis, as follows. The same notation used in Section 3.1 is applied, so let $\texttt{lChoice}(\mathcal{B})$ be the set of signed left-choice formulas in branch $\mathcal{B}$; consider a partition of this set, $\texttt{lChoice}(\mathcal{B}) = \Lambda^T \cup \Lambda^F$, where $\Lambda^T = \{A \mid TA \in \texttt{lChoice}(\mathcal{B})\}$ and $\Lambda^F = \{A \mid FA \in \texttt{lChoice}(\mathcal{B}_i)\}$. Define the formula $\lambda_{\mathcal{B}}$:

$$\lambda_{\mathcal{B}} = \begin{cases} F \bigwedge \Lambda^T, & \text{if } \Lambda^F = \varnothing \\ T \bigvee \Lambda^F, & \text{if } \Lambda^T = \varnothing \\ T\left(\bigwedge \Lambda^T\right) \rightarrow \left(\bigvee \Lambda^F\right) & \text{otherwise} \end{cases}$$

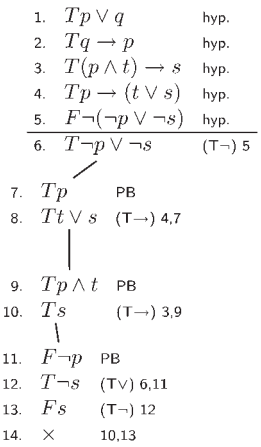


FIG. 9. Left-most closed branch in Figure 7

Note that $\lambda_{\mathcal{B}}$ is precisely the right branching formula computed in the comb normal form for branch $\mathcal{B}$ in Theorem 3.1.

This is perhaps better understood by means of an example. Consider a depth-first construction of the tableau proof in Figure 7. Consider the proof when it reaches the first closed branch, i.e. the left-most closed branch. This situation is illustrated in Figure 9.

The cut formulas in this tableau are the decisions made in the search of an open branch, namely $Tp, Tp\wedge t$ and $F\neg p$. With this point of view, this was a set of bad decisions that should be avoided. That is, if Tp and $Tp\wedge t$ are to be found in a branch, than one *must have* $T\neg p$ in it. This corresponds to learning $T(p\wedge(p\wedge t))\rightarrow\neg p$. Note that this is exactly the formula present in the right branch of the first use of PB in Figure 8 and, similarly, the left branch corresponds to the left-most branch of Figure 8. If we add this learned formula to the top of the tableau, one can then proceed the search with that extra formula added, as illustrated in Figure 10.

The second closed branch is basically the same as in Figure 7. It must be noted that the right PB formula $T\neg p$ can, in fact, be inferred from the learned formula $T(p\wedge(p\wedge t))\rightarrow\neg p$ and the remaining choice formulas Tp and $Tp\wedge t$ with two successive applications of $(T\wedge\rightarrow)$. This is due to the fact that, in the presence of the choice formulas, the learned formula is equivalent to the right PB formula.

At this point, we know that the choice formulas Tp and $Tp\wedge t$ did not lead to an open branch, so those two labelled formulas should not occur again in a branch. This statement implies the learning of $Fp\wedge(p\wedge t)$; of course, one could simplify and simply learn $Fp\wedge t$, but we will not do it here. Note that $Fp\wedge(p\wedge t)$ is the second right PB formula in Figure 8, which is no coincidence since the method for obtaining both are exactly the same.

We could go on and add the learned formula $Fp\wedge(p\wedge t)$ to the top of the tableau, but not much would be changed in the original proof of Figure 7. There is a repeated pattern here, in which the learned formulas are basically the same as the right PB formulas in Figure 8. If every time a new formula is learned we would restart the tableau from the beginning, but maintaining all learned formulas, the proof obtained by this successive restarts would be exactly the comb-tree proof of Figure 8. For example, if instead of proceeding to the right branch in Figure 10, we had restarted the tableau, using the top learned formula

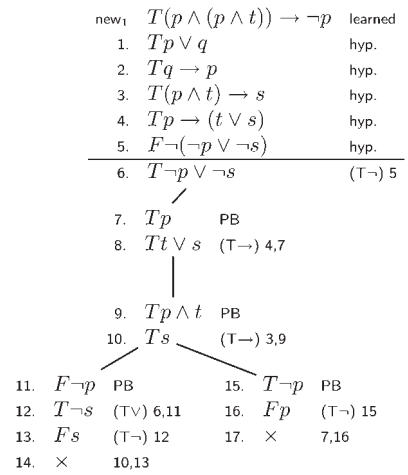


FIG. 10. Next left-most closed branch in Figure 7

$T(p\wedge(p\wedge t))\rightarrow\neg p$, we could branch over $Tp\wedge(p\wedge t)$, thus generating $T\neg p$, which is exactly what happens in the comb tree proof of Figure 8. In this sense, we can consider this learning expansion as an on-line construction of the comb-proof.

The truth is, not much is gained by this form of learning, due to the equivalence between learned formulas and right PB formulas pointed above. In fact, let $\psi_1,\ldots,\psi_n$ be an enumeration of the left choice formulas in a branch $\mathcal{B}$; then, in the presence of $\psi_1,\ldots,\psi_{n-1}$, $\lambda_{\mathcal{B}}$ and the learned right PB formula (which we called $\bar{\lambda}_n$) are equivalent. This means that when $\mathcal{B}$ closes, adding $\lambda_{\mathcal{B}}$ to the set of premisses provides exactly the same information as $\bar{\lambda}_n$, which is next right choice formula to be added to the first remaining open branch.

It is no surprise that the first SAT-solvers to deal with the learning of clauses reported worse efficiency results when learning was enabled.

However, there is a practical application of this form of learning in the presence of *random restarts*. This is a technique applied in many SAT-solvers (e.g. in Chaff [MMZ+01]) such that when a branch closes there is a probability of the process of theorem proving restarting. This does not mean that all the work done so far is lost, precisely because the formulas learned are not deleted at restart. The restart allows the prover to recover from bad initial branching choices, keeping all the learned knowledge so far. This remains true even for other kinds of learning, one of which we detail next.

4.2 Inference-based Learning

In general, SAT-solvers use clausal representation, so that at each conflict detection, i.e. at branch closing, a new clause is inferred. A broad investigation on several methods for clause learning based on *implication graphs* can be found in [ZMMM01], in which empirical tests have indicated that the learning technique generating smaller proofs is *resolution-based*, where the learned clause is obtained by resolving a set of clauses involved in the generation of the conflict. In the simpler case, the two clauses that contain the pair of literals that generated the conflict are resolved; more sophisticated methods are presented in [ZMMM01]

TABLE 2. Signed Formulas Learned by General Resolution at Branch Closing

Origin of TX	*Origin of FX*	*Learned Formula*	*Justification*
$T\ A[X]$	$T\ B[X]$	$T\ A[X/\bot]\vee B[X/\top]$	$A[X], B[X]\vdash A[X/\bot]\vee B[X/\top]$
$T\ A[X]$	$F\ B[X]$	$T\ B[X/\top]\rightarrow A[X/\bot]$	$A[X]\vdash B[X], B[X/\top]\rightarrow A[X/\bot]$
$F\ A[X]$	$T\ B[X]$	$T\ A[X/\bot]\rightarrow B[X/\top]$	$B[X]\vdash A[X], A[X/\bot]\rightarrow B[X/\top]$
$F\ A[X]$	$F\ B[X]$	$F\ A[X/\bot]\wedge B[X/\top]$	$A[X/\bot]\wedge B[X/\top]\vdash A[X], B[X]$

That method is resolution based, and hence clausal. However, it can be generalised to any formula by using Bachmair-Ganzinger *general resolution* rule [BG01]:

$$\frac{A[B] \qquad A'[B]}{A[B/\bot]\vee A'[B/\top]}$$

where $X[Y]$ designates the formula X in which Y is a subformula, and $X[Y/Z]$ designates the same formula X with all occurrences of Y replaced by Z.

Now suppose we are in a branch that closes due to the presence of TX and FX, and that at this moment a new formula is to learned using general resolution. As all connective rules are analytic, we can trace back the origin of TX and FX either to an original hypothesis or to a PB (decision) formula; if PB is analytic, then TX and FX can always be traced back to some hypothesis, respectively designated as $A[X]$ and $B[X]$. The formula learned is the result of general resolution applied to signed formula, as indicated in Table 2.

The justification column in Table 2 is always an instance of the general resolution principle, adapted to the tableau rules. In fact, the generalised tableau rule requires that both hypothesis be positive, that is T-marked. This only occurs in line 1. In lines 2–4, some classical transformation was applied. For instance, in line two the hypotheses are $T\ A[X]$ and $F\ B[X]$, which is classically equivalent to having $T\ A[X]$ and $T\ \neg B[X]$; so if we applied general resolution to the latter hypotheses, we would obtain $T\ A[X/\bot]\vee\neg B[X/\top]$, which is classically equivalent to $T\ B[X/\top]\rightarrow A[X/\bot]$, as desired. Lines 3 and 4 of Table 2 were obtained analogously.

In Table 2, the justification sequent always contains the origins of TX and FX and also the *opposite* of the leaned formula. By assuming this justification sequent as a valid sequent, we can derive a new inference rule which allows us to add the learned formula to the end of an expanding branch whenever it contains particular point of the tableau. For instance, the sequent in the justification column of the first line of Table 2 is $A[X], B[X]\vdash A[X/\bot]\vee B[X/\top]$, implies that the following tableau always closes:

1.	$TA[X]$	hyp.
2.	$TB[X]$	hyp.
3.	$FA[X/\bot]\vee B[X/\top]$	general resolution
	$\times$	

In fact, the general resolution is a valid rule, so by completeness we know that the tableau above always closes, but by considering general resolution as a derived rule so be selectively using at branch closing, we may be able to generate smaller proofs. In this way, the closed

tableau above can be used to derive the inference rule $(GR)_1$

$$\frac{\begin{array}{c} TA[X] \\ TB[X] \end{array}}{TA[X/\bot]\vee B[X/\top]}\,(\mathrm{GR})_1$$

due to the following tableau, whose left branch is precisely the closed tableau above:

$$\begin{array}{ccc} & TA[X] & \\ & TB[X] & \\ FA[X/\bot]\vee B[X/\top] & & TA[X/\bot]\vee B[X/\top] \\ \times & & \end{array}$$

Similarly, we can derive rules $(GR)_2$,$(GR)_3$ and $(GR)_4$, corresponding to the other lines of Table 2. When we use those derived inference rules, we cannot always add the learned formula to the root of the tableau, for the resolvents (that is, the origins of TX and FX) must be present in the branch. The insertion point of the learned formula is just after the lowest of $A[X]$ and $B[X]$ in a tableau branch, as shown in Figure 11 for the first case in Table 2.

Figure 12 presents the tableau of Figure 7 transformed with resolution-based learning of formulas at branch closing.

The boxed formulas are the learned formula. Formula (1) was learned at the closing of the branch 1, with $A[s]=T(p\wedge t)\rightarrow s$ and $B[s]=T\neg p\vee\neg s$. Formula (2) was learned at the closing of branch 2, with $A[p]=Tp$ and $B[p]=T\neg p\vee\neg s$. No formula was added at the closing of branch 3, although this is also possible. Some trivial simplifications were employed in Figure 12 so as to eliminate $\top$ and $\bot$ from the learned formulas.

Note that the addition of the learned formula is made *a posteriori*, that is, the use of the derived inference is made only after the branch closes. So formula (1) was not present when branch 1 closes, as is indicated by the node number 15. After branch 1 closes (node 14) and before backtracking to expand branch 2 (node 16), formula (1) is added at that precise point in the tableau, and it remains available for further expansions. In this particular case, it was inserted at a point high enough so that it remains available for all further expansions. This is not the case of the learned formula (2), which was inserted at node 19 after branch 2 closes at node 18. Formula (2) was available to branch the expansion of branch 3, and in fact, it was used there to derive Fs (node 21), but due to its insertion position, it was not available to the expansion of branch 4.

$$\begin{array}{l} \vdots \\ T\ B[X] \\ \vdots \\ TA[X] \\ \quad \bullet\ \leftarrow \text{insert } T\ A[X/\bot]\vee B[X/\top] \text{ here} \\ \vdots \\ \times \end{array}$$

FIG. 11. Insertion point of resolution-based learning

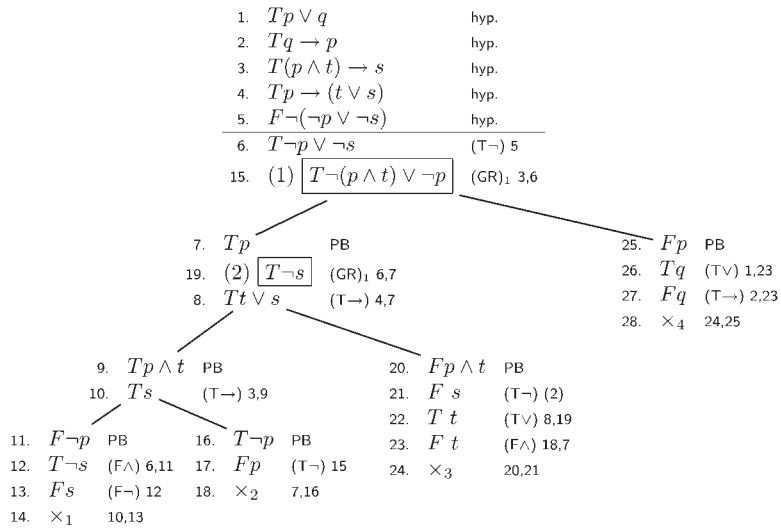


FIG. 12. Proof of sequent using inferential learning

Note that formula (1) was not employed in the proof, but formula (2) enabled the closing of branch 3, thus reducing the number of branches and formulas in the proof. If we compare Figure 12 with the proof of Figure 7 of the same sequent, we see that the learning of formula (2) allowed us to avoid some extra branching, and in fact reduced the size of the proof.

In this respect, we see that the learned formulas are always of size polynomial with the original hypothesis, and are added at most one for each branch, so that the tableau using learned formulas is at most polynomially larger (in the number of symbols) than the tableau without the use of this form or learning. As seen in the example, the tableau with learning has the potential of having fewer branches, and thus of generating smaller proofs.

LEMMA 4.1
Adding the formulas learned according to Table 2 generates only sound proofs.

PROOF. Suppose we want to prove $\Gamma \vdash \Delta$. Suppose we have a closed branch that adds a new formula. Without loss of generality, suppose the first line of Table 2 was used. As usual with tableau reading of sequent rules, we have to read rules bottom-up.

Consider initially the case where the formulas that cause the closure of the branch can be traced back to the hypothesis; in this case, let $\Gamma = \Gamma', A[X], B[X]$, so that the sequent to be shown is $\Gamma', A[X], B[X] \vdash \Delta$. By adding the learned formula and closing the tableau, we have actually proved that $\Gamma', A[X], B[X], A[X/\bot] \vee B[X/\top] \vdash \Delta$. Then, with a cut application with the justification sequent $A[X], B[X] \vdash A[X/\bot] \vee B[X/\top]$, which is a simple application of general resolution, we obtain $\Gamma', A[X], B[X] \vdash \Delta$ as desired.

Now suppose that when we closed the branch, the formulas that caused the closure cannot be traced back to Γ or Δ, but trace back to some branching formula. However, the same reasoning as above applies. When the KE branch is closed, we have actually proved

$$\Gamma, C_1, \ldots, C_n \vdash D_1, \ldots D_m, \Delta,$$

where C_i are the T-marked branching formulas and D_j are the F-marked branching formulas

in the branch. Suppose the last PB was over D_m so that closing the next branch would be equivalent to proving

$$\Gamma, C_1, \ldots, C_n, D_m \vdash D_1, \ldots D_{m-1}, \Delta.$$

However, as we assume that $(GR)_1$ is applicable, $A[X], B[X] \in \{\Gamma, C_1, \ldots, C_n\}$, so that we end up proving

$$\Gamma, C_1, \ldots, C_n, D_m, A[X/\bot] \vee B[X/\top] \vdash D_1, \ldots D_{m-1}, \Delta.$$

Again, with a cut with justification $A[X], B[X] \vdash A[X/\bot] \vee B[X/\top]$, we obtain the desired sequent.

The other three cases are totally analogous. We omit the details. ∎

The use of extra tableau inference rules $(GR)_{1-4}$ is not restricted to KE-type of tableaux. In fact, the same enhancement is possible, for instance, in analytic tableaux. However, unlike analytic tableaux, the use of those rules can be polynomially simulated in KE-tableaux.

THEOREM 4.2
Inference-based learning corresponds to the employment of non-analytic branchings in proofs.

PROOF. First note that the learned formulas in Table 2 are not, in general, a subformula of any other formula in the tableau, so that if there is a branching over one of those formulas, the proof is, in general, non-analytic.

So, if the premisses of one of $(GR)_{1-4}$ is present in a partially expanded branch $\mathcal{B}$, we can branch over the learned formula, generating $\mathcal{B}_l$ and $\mathcal{B}_r$; suppose the learned formula is on $\mathcal{B}_r$.

Due to the validity of general resolution and the completeness of KE-tableaux, the left branch $\mathcal{B}_l$ closes.

The expansion of $\mathcal{B}_r$ proceeds as normally, and now it has an added formula, which increases the chances of new inferences and thus of closing the branch. ∎

The peculiarity of this non-analytic branching is that it is an *a posteriori* branching only added after the branch is closed. In practice, this a posteriori branching does not occur, and is used only to justify the *a posteriori* addition of an extra formula at a particular place. When the learned formulas is added at the root of the tableau, this corresponds to an application of PB at the very opening of the tableau.

Finally, we present in Figure 13 another version of the tableau in Figure 12 in which all non-analytic branchings are presented explicitly.

Figure 13 indicates how the learned formulas can be proved, but the rest of the tableau, being identical to Figure 12, is omitted. It is no coincidence that in Figure 13 the left branches corresponding to learned formulas (1) and (2) close in the same fashion that branches 1 and 2 in Figure 12. In this sense, the use of the derived rules based on general resolution $(GR)_{1-4}$ avoid duplication.

It is clear that using $(GR)_{1-4}$ only a polynomial number of formulas (of polynomially bounded size) can be added to the original proof, so the size of the transformed proof is polynomially bounded on the size of the proof without learning. Furthermore, the addition of the learned formula has the potential of actually reducing the size of the original proof, i.e. the proof without employing learning.

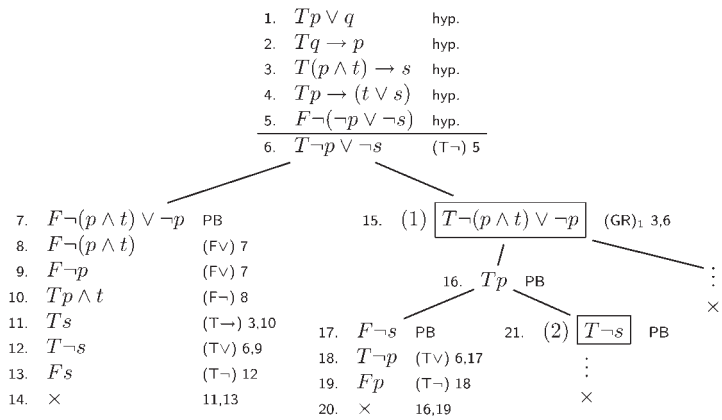


FIG. 13. Proof of sequent deriving learned formulas

We have thus shown that one possible way of computing non-analytic branchings is by learning formulas using generalised resolution at branch closing.

5 Conclusion

We have opened this paper by stating that it is a myth the usually accepted statement that computing non-analytic cuts in sequent proofs is hard.

To achieve this goal, we have provided a way of transforming any cut-free, analytic or non-analytic proof into a normal form non-analytic proof, such that the transformed proof is polynomially bounded by the size of the original proof. This normal form consisted of a degenerated comb-tree proof with the same number of branches as the original formula. We have also shown that such proof can be built during proof construction by using decision-based learning.

We have also provided another method based on generalised resolution that computes (a posteriori) non-analytic cuts during proof construction based on the learning formulas from closed branches.

Several future paths of research are indicated by the present work. One is actually an appreciation of how much proofs are shrunk, if at all, using the learning technique described above.

Another path of research would be to try to obtain other methods for computing non-analytic cuts, for the methods described here by no means exhaust the subject.

The computation of non-analytic cuts for other logics (modal, first-order and non-classical logics) is also a possibility. Possibly, many techniques already employed by practical theorem provers can be rendered as non-analytic cuts. One candidate in this direction is *backjumping*, a search pruning technique employed in a variety of areas.

Acknowledgements

Marcelo Finger is partly supported by CNPq grant PQ 301294/2004-6 and FAPESP project 04/14107-2.

References

[BG01] L. Bachmair and H. Ganzinger. Resolution theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 2, pages 19–99. Elsevier Science, 2001.

[BM96] R. J. Bayardo and D. P. Miranker. A complexity analysis of space-bounded learning algorithms fofr the constraint satisfaction problem. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 508–522, 1996.

[Boo84] George Boolos. Don't eliminate cut. *Journal of Philosophical Logic*, 13:373–378, 1984.

[CF58] H. B. Curry and R. Feys. *Combinatory Logics*, volume 1. North Holland, 1958.

[CS00] Alessandra Carbone and Stephen Semmes. *A Graphic Apology for Symmetry and Implicitness*. Oxford Mathematical Monographs. Oxford University Press, 2000.

[D'A90] Marcello D'Agostino. Investigations into the complexity of some propositional calculi. Technical Report PRG Technical Monograph 88, Oxford University Computing Laboratory, 1990.

[D'A92] Marcello D'Agostino. Are tableaux an improvement on truth-tables? — Cut-free proofs and bivalence. *Journal of Logic, Language and Information*, 1:235–252, 1992.

[D'A99] Marcello D'Agostino. Tableau methods for classical propositional logic. In Marcello D'Agostino, Dov Gabbay, Rainer Haehnle, and Joachim Posegga, editors, *Handbook of Tableau Methods*, pages 45–124. Kluwer, 1999.

[Dec90] Rina Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41(3):273–312, January 1990.

[DLL62] M. Davis, G. Longemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[DM94] Marcello D'Agostino and Marco Mondadori. The taming of the cut. Classical refutations with analytic cut. *Journal of Logic and Computation*, 4(285–319), 1994.

[FG06] Marcelo Finger and Dov Gabbay. Cut and pay. *Journal of Logic, Language and Information*, 15(3):195–218, October 2006.

[Fit83] Melvin Fitting. *Proof Methods for Modal and Intuitionistic Logic*. Reidel, 1983.

[GN02] E. Goldberg and Y. Novikov. Berkmin: A Fast and Robust SAT Solver. In *Design Automation and Test in Europe (DATE2002)*, pages 142–149, 2002.

[How80] W. Howard. The formulae-as-types notion of constructuion. In J. R. Hinchley and J. P. Seldin, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[Kle67] Stephen C. Kleene. *Mathematical Logic*. John Wiley & Sons, 1967.

[Mas00] Fabio Massacci. Single step tableaux for modal logics: Methodology, computations, algorithms. *Journal of Automated Reasoning*, 24(3):319–364, 2000.

[ML79] P. Martin-Löf. Constructive mathematics and computer programming. In *Logic, Methodology and Philosophy of Science*, volume VI, pages 153–175. North Holland, 1979.

[MMZ+01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)*, pages 530–535, 2001.

[MSS99] J. P. Marques-Silva and K. A. Sakallah. Grasp: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48:506–521, 1999.

[NOT05] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Abstract DPLL and abstract DPLL modulo theories. In F. Baader and A. Voronkov, editors, *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'04), Montevideo, Uruguay*, volume 3452 of *Lecture Notes in Computer Science*, pages 36–50. Springer, 2005.

[Smu68a] Raymond M. Smullyan. Analytic Cut. *Journal of Symbolic Logic*, 33:560–564, 1968.

[Smu68b] Raymond M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.

[Sza69] M. E. Szabo, editor. *Collected papers of Gerhard Gentzen*. Studies in Logic, Amsterdam, 1969.

[ZMMM01] L. Zhang, C. F. Madigan, M. H. Moskewitz, and S. Malik. Efficient conflict driven learning in a boolean satisfiability solver. In *International Conference on Computer-Aided Design (ICCAD2001)*, 2001.