Cut-Based Abduction

Marcello D'Agostino Marcelo Finger[*] Dov Gabbay

Abstract

In this paper we explore a generalization of traditional abduction which can simultaneously perform two different tasks: (i) given an unprovable sequent $\Gamma \vdash G$, find a sentence H such that $\Gamma, H \vdash G$ is provable (hypothesis generation); (ii) given a provable sequent $\Gamma \vdash G$, find a sentence H such that $\Gamma \vdash H$ and the proof of $\Gamma, H \vdash G$ is simpler than the proof of $\Gamma \vdash G$ (lemma generation). We argue that the two tasks should not be distinguished, and present a general procedure for finding suitable hypotheses or lemmas. When the original sequent is provable, the abduced formula can be seen as a cut formula with respect to Gentzen's sequent calculus, so the abduction method is cut-based. Our method is based on the tableau-like system KE and we argue for its advantages over existing abduction methods based on traditional Smullyan-style Tableaux.

1 Introduction

The aim of this paper is to outline a method for solving the following problem: given a sequent $\Gamma \vdash G$ whose validity is yet undecided, produce a sentence H, such that:

- $\Gamma, H \vdash G$ is provable;
- if $\Gamma \cup \{G\}$ is consistent then $\Gamma \cup \{H\}$ is consistent; and $H \nvdash G$, that is, H alone does not imply G.

The idea is that Γ is a contextual database, containing background knowledge, G is a goal formula representing some fact or evidence that we want to explain or prove, and H is a sentence which represents either an extra hypothesis necessary to explain the evidence G or a "lemma", namely a sentence that follows from the background knowledge Γ, but allows for an easier proof of G. This is quite similar to what has been usually regarded as falling in the scope of *abductive reasoning*, since its introduction by Peirce [18]. Usually, this activity involves the generation of hypotheses and the choice of the best hypotheses[1]. In this context, the first condition above requires that the produced hypothesis, together with the database Γ, explains the evidence. The second condition does not imply a deterministic choice of what constitutes a "best" hypothesis, but avoids trivial answers; that is, it avoids the generation of an abduced hypothesis that either makes the antecedent inconsistent when the input was consistent, or contains the evidence.

Our task, however, is *not exactly* the usual abductive reasoning found in the literature, for the latter requires the goal formula G not to be derivable from the background knowledge, ie $\Gamma \nvdash G$ [20, 19, 21, 13, 8, 1, 16]. It can rather be seen as a generalisation of the traditional abduction task, simultaneously covering two cases:

(a) if $\Gamma \nvdash G$, the problem reduces to traditional abduction, which we call *hypothesis generation*. In traditional abduction, an explanation has to be produced to turn an originally falsifiable sequent into a valid one. Often some notion of minimality or preference is

[*]Partly supported by CNPq grant PQ 301294/2004-6 and FAPESP project 04/14107-2.

[1]"Abduction consists in studying facts and devising a theory to explain them." (C. P. Peirce, Collected Papers, vol. V, par. 145.) "Abduction is the process of forming an explanatory hypothesis. It is the only logical operation which introduces any new idea" (C. P. Peirce, Collected Papers, vol. V, par. 171).

Vol. 16 No. 6, © The Author 2008. Published by Oxford University Press. All rights reserved.
For Permissions, please email: journals.permissions@oxfordjournals.org
doi:10.1093/jigpal/jzn020

involved in choosing the best explanation [20, 19, 8]. Here we will search for a compromise between minimality and computational efficiency.

(b) if $\Gamma \vdash G$ is provable, the task is not that of explaining a given set of data, but that of facilitating its proof, which we call *lemma generation*. We further expect the produced provable sequent $\Gamma, H \vdash G$ to have a simpler proof than $\Gamma \vdash G$, where "simpler" may mean "shorter" or just "easier to grasp". There is also a compromise to be reached between finding a simpler proof and finding a proof at a small computational cost.

In many cases, one adds a hypothesis H which is not provable from the data, but which facilitates the proof of provable theorems from the data. This can be found in several texbooks for mathematics (algebra, calculus, etc), in which extra hypotheses are added to provable theorems just to make them more adequate to the background of students. Although this is not quite the case of lemma generation, it illustrates that it is not so uncommon that extra hypotheses are added to provable theorems just to obtain simpler proofs. A deeper discussion on these matters can be found in [15].

So, we are expanding the traditional *explanationist* view of abduction, with a capacity of providing a simpler or more compact account of facts, as present in the desiderata of [16]. We call this a *simplificationist* view of abduction.

In this latter case, the abduced hypothesis H can be seen as the computation of a *cut formula*. In fact, once we have efficiently proved $\Gamma, H \vdash G$, we can start looking for an efficient proof of $\Gamma \vdash G$, by searching for a deduction of H from Γ as an auxiliary lemma, that is trying to show that $\Gamma \vdash H$. If an efficient proof of this lemma is found, we can put the proofs together via a cut inference:

$$\dfrac{\dfrac{\Gamma \vdash H \quad \Gamma, H \vdash G}{\Gamma, \Gamma \vdash G}\ (\mathit{Cut})}{\Gamma \vdash G}\ (\mathit{Contraction})$$

and thus obtain an efficient proof of $\Gamma \vdash G$. For a potentially smaller proof, instead of proving $\Gamma \vdash H$, one could prove $\Gamma \vdash H, G$ such that, after cut, G would be contracted on the right of $\vdash$ as well, leading to the desired $\Gamma \vdash G$.

There may be advantages in proving $\Gamma \vdash H, G$ instead of $\Gamma \vdash H$. In fact, while the former sequent has one extra formula in the succedent, in any refutation method its proof is no longer than that of the latter, nor is it longer than that of the original sequent $\Gamma \vdash G$.

One of the possible ways to satisfy our goals in case (b) is to start with a proof search for $\Gamma \vdash G$, for example, a cut free or analytic proof, and generate a more efficient proof, possibly with non-analytic cuts, for it is known that proofs with non-analytic cuts can be exponentially smaller than cut-free and analytic proofs [2, 3, 4]. When lemma generation is successfully used to find shorter proofs we say we have *abduced proof efficiency*. We stress that the generalised abduction method developed here does not require any a priori knowledge of whether $\Gamma \vdash G$ or not.

The wider view of abduction we propose here can therefore play a twofold role. We may not know in advance which role it is playing, because we may not know in advance whether the sequent is provable or not. This is what actually happens in scientific reasoning. We may find an abduced hypothesis H, and then discover that H is a "lemma", that is, a theorem that we have already proven or may be easier to prove. So, we may not know, in advance, whether we are simply improving on the efficiency of the proof via a suitable lemma, or we are actually adding a genuine new hypothesis which is required to prove the goal.

The use of analytic tableaux for abduction has been presented in the literature for several logics [6, 7, 9]. However, as pointed out above, the cut rule plays a special role in this problem, so we do not use tableau methods based on cut-free sequent calculus, such as Smullyan's Analytic Tableaux [22], but employ instead the KE tableau method which is able to efficiently simulate sequent calculus with full use of cuts [12]. The abduction procedure we propose here can be directly applied during a KE-tableau expansion, without prior knowledge of whether the input sequent is indeed provable or not. Roughly speaking, it consists of generating, for each open branch $\mathcal{B}$ of a KE-tableau for a sequent $\Gamma \vdash G$, a sentence $H_{\mathcal{B}}$ whose addition to the background knowledge Γ is sufficient to close that particular branch; we call this method *branch-driven abduction*.

When used as a lemma generation method, branch driven abduction cannot be applied statically to a completed tableau, but must be applied dynamically to an uncompleted one in order to generate hopefully smaller proofs on-the-fly. So, we present a *dynamic abduction* algorithm and discuss the structure of the proofs thereby generated. We then prove the correctness of the algorithm and discuss its termination.

Throughout the paper, the discussion of abduction methods and proof-size reduction is performed in the context of classical propositional logic. We believe that those techniques can be applied to other logics, such as first-order and modal logics, with additional effort to deal with quantifiers and modalities; it may be also interesting to see the applications of these ideas to abduction over non-classical logics, such as in [5]. But these extensions are left out of the scope of this paper.

It is important to note that we do not claim that *all* propositional theorems can have a small proof in this way, otherwise we would have proved that NP=coNP, which we have not. However, with this work, abduction remains *one possible technique* for proof compression.

The rest of the paper is organised as follows. Section 2 presents some preliminary definitions and results used in the rest of the paper. The KE-based method called *branch-driven abduction* is presented in Section 3. In Section 4 we compare our KE-based method with another one based on traditional (Smullyan-style) tableaux, showing that a very desirable feature of the former, namely the logical independence of the hypotheses abduced from distinct branches, is not shared by the latter. Finally, the use of our KE-based method for "lemma generation" via a *dynamic abduction* procedure is discussed in Section 5. In the same section we discuss how one can make advantage of this approach in computing (possibly shorter) non-analytic proofs. Concluding remarks and pointers to further work are presented in Section 6.

2 Preliminaries

We consider formulas built over a countable set of propositional atoms denoted by the lower case letters p, q, r, etc., and connectives $\neg$, $\wedge$, $\vee$ and $\rightarrow$. We represent formulas by upper case Latin letters: A, B, C, etc. We represent sets of formulas by upper case Greek letters, such as Γ, Δ, Φ and Ψ. We write Γ, A to represent $\Gamma \cup \{A\}$ and Γ, Δ to represent $\Gamma \cup \Delta$.

A *sequent* is an expression of the form $\Gamma \vdash \Delta$, where Γ is the antecedent and Δ is the succedent. A sequent inference rule allows one to infer a sequent $\mathcal{S}$ from zero or more sequents $\mathcal{S}_1, \ldots, \mathcal{S}_k$. If the inference rule has 0 premisses, it is called a sequent *axiom*. A k-premissed

Table 1. Smullyan's unifying notation

α	α_1	α_2
$TA\wedge B$	TA	TB
$FA\vee B$	FA	FB
$FA\rightarrow B$	TA	FB
$T\neg A$	FA	FA
$F\neg A$	TA	TA

β	β_1	β_2
$TA\vee B$	TA	TB
$FA\wedge B$	FA	FB
$TA\rightarrow B$	FA	TB

inference rule, with $k>0$, is called *analytic* if every formula in the k premisses occurs as a subformula in the conclusion $\mathcal{S}$. In most sequent calculi, all rules are analytic, except for the *cut rule*, which we assume to have the following format

$$\frac{\Gamma_1\vdash\Delta_1, A \qquad A,\Gamma_2\vdash\Delta_2}{\Gamma_1,\Gamma_2\vdash\Delta_1,\Delta_2}\;(Cut)$$

The formula A is called the *cut formula* of this inference and the cut is analytic if A occurs in $\Gamma_1,\Gamma_2\vdash\Delta_1,\Delta_2$. A sequent proof is a tree whose nodes are sequents, having sequent axioms at its leaves and such that every internal node is obtained by the application of some k-premissed inference rule, $k>0$. A proof is analytic if it only uses analytic inferences. We assume that the only potentially non-analytic rule is cut, so that a cut-free proof is always analytic. We write $\Gamma\vdash^{?}\Delta$ when we do not know if $\Gamma\vdash\Delta$ is provable.

In the sequel, we shall often find it convenient to use Smullyan's unifying notation for non-atomic signed formulas described in Table 1.

2.1 KE Tableaux

KE-tableaux ([12], see also [10, 11]) were proposed by D'Agostino and Mondadori as a way of incorporating in a non-redundant way the cut rule into tableau proofs. This rule, they argue, corresponds to the principle of excluded middle or, at the semantic level, to the *principle of bivalence* (PB) which is essential of efficient proof-theoretical treatment of classical logic. Furthermore, it has been shown that, in comparison with Smullyan's analytic tableau [22], based on the cut-free sequent calculus, KE-tableau have better computational properties and that the use of the cut rule can be suitably restricted so as to preserve all desirable properties of cut-free proofs. The original presentation of KE-tableaux was based on considerations of non-redundancy over Kleene's **G4** sequent system and Smullyan's analytic tableaux. It was also shown that the same tableau system can be obtained using a variant sequent calculus called the cut-based calculus [14].

KE-tableaux deal with *signed* formulas. If A is a formula, $T\ A$ and $F\ A$ are signed formulas. $T\ A$ is the *conjugate formula* of $F\ A$, and vice versa; if $X\in\{T,F\}$ then $\bar{X}$ is defined as follows: $\bar{T}=F$ and $\bar{F}=T$. Each connective is associated with a set of *linear expansion rules* also called *elimination rules*. Linear expansion rules always have a *main premiss*, i.e. the one containing the connective to be eliminated; two-premiss rules also have an *auxiliary premiss*. Figure 1 shows the KE linear expansion rules for classical logic. 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 2, which corresponds to the cut rule. A KE-tableau for the sequent $A_1,\ldots,A_n\vdash B_1,\ldots,B_m$ starts, like a standard tableau, 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

$$
\begin{array}{ccc}
\dfrac{T\ A \wedge B}{\begin{array}{c}T\ A\\ T\ B\end{array}}\ (T\wedge) &
\dfrac{\begin{array}{c}F\ A \wedge B\\ T\ A\end{array}}{F\ B}\ (F\wedge_1) &
\dfrac{\begin{array}{c}F\ A \wedge B\\ T\ B\end{array}}{F\ A}\ (F\wedge_2) \\[2ex]
\dfrac{\begin{array}{c}T\ A \vee B\\ F\ A\end{array}}{T\ B}\ (T\vee_1) &
\dfrac{\begin{array}{c}T\ A \vee B\\ F\ B\end{array}}{T\ A}\ (T\vee_2) &
\dfrac{F\ A \vee B}{\begin{array}{c}F\ A\\ F\ B\end{array}}\ (F\vee) \\[2ex]
\dfrac{\begin{array}{c}T\ A \rightarrow B\\ T\ A\end{array}}{T\ B}\ (T\rightarrow_1) &
\dfrac{\begin{array}{c}T\ A \rightarrow B\\ F\ B\end{array}}{F\ A}\ (T\rightarrow_2) &
\dfrac{F\ A \rightarrow B}{\begin{array}{c}T\ A\\ F\ B\end{array}}\ (F\rightarrow) \\[2ex]
\dfrac{T\ \neg A}{F\ A}\ (T\neg) &
\dfrac{F\ \neg A}{T\ A}\ (F\neg) &
\end{array}
$$

FIG. 1. KE Expansion Rules

$$\begin{tikzcd}
& {} \arrow[dl, dash] \arrow[dr, dash] & \\
T\ A & & F\ A
\end{tikzcd}$$

FIG. 2. Principle of Bivalence

of an expansion rule are present in the branch; the branch expansion consists of adding the conclusions of the rule to the end of that branch (one can simultaneously expand all branches passing through the premisses of that rule). The PB branching rule splits a branch into two. The final goal of tableau expansion is to produce what is called a *completed* KE-tableau. Before giving a proper definition of a completed KE-tableau, we first introduce some useful terminology.

A signed formula is *fulfilled* in a branch if it is the main formula of an instance of an expansion rule such that the conclusions of that instance already occur in the branch. Clearly, if a formula is the main formula of an expansion it becomes fulfilled; however, it is possible for a non-atomic signed formula to become fulfilled by the expansion of other signed formulas as well. Note that any valuation that satisfies the conclusions of an instance of an expansion rule also satisfies its main formula, so that fulfilled formulas are "subsumed" by other formulas in the branch. Note that, using Smullyan's unifying notation (see Section 2 above), our definition of a fulfilled signed formula can be restated as follows: an α is fulfilled in a branch $\mathcal{B}$ if and only if both α_1 and α_2 occur in $\mathcal{B}$; a β is fulfilled in $\mathcal{B}$ if and only if either β_1 or β_2 occur in $\mathcal{B}$. By definition, a signed atomic formula is always unfulfilled.

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. A branch is *saturated* if it is open and all non-atomic signed formulas in it are fulfilled and *completed* if it is either closed or saturated. Finally, the tableau is *completed* if every branch in it is completed.

The deductibility relation $\vdash_{\mathbf{KE}}$ is defined as follows:

$$A_1, \ldots, A_n \vdash_{\mathbf{KE}} B_1, \ldots, B_m \text{ iff there is a closed KE-tableau for } TA_1, \ldots, TA_n, FB_1, \ldots, FB_m. \quad (1)$$

Any saturated branch provides a counter-valuation for the sequent.

Although the applications of PB may introduce arbitrary signed formulas, so violating the subformula property, D'Agostino and Mondadori [12] showed that there is no loss of completeness in restricting the applications of PB in such a way as to preserve the subformula property.

A typical analytical KE-procedure, described in terms of Smullyan's unifying notation, is outlined in Algorithm 2.1. Note that, as branching is potentially explosive, one typically applies the only branching rule of KE (PB) only after all linear expansions have been applied.

Algorithm 2.1 The typical (analytical) KE-procedure

KEProof(Γ, G)

Input: A sequent $\Gamma \vdash^? G$,

Output: A completed KE-tree.

```
 1: Construct a one-branch tree containing all the signed formulas in
    $\{TA : A \in \Gamma\} \cup \{FG\}$ (in any order).
 2: while the tree is uncompleted do
 3:   choose an uncompleted branch $\mathcal{B}$
 4:   while $\mathcal{B}$ is uncompleted do
 5:     if $\mathcal{B}$ contains the premisses of a linear expansion rule, but not its conclusion(s) then
 6:       apply the relevant linear expansion rule to $\mathcal{B}$ and append its conclusion(s) to the
          end of $\mathcal{B}$
 7:       /* Some $\beta$ may remain unfulfilled even when all the
          possible linear expansion rules have been applied */
 8:     else if $\mathcal{B}$ contains an unfulfilled $\beta$ then
 9:       /* This is an application of PB */
10:       append $\beta_1$ and $\bar{\beta}_1$ as sibling nodes to the end of $\mathcal{B}$
11:     end if
12:   end while
13: end while
```

Fact 2.1 *The procedure* KEProof *is sound and complete w.r.t. classical logic.*

As an example, Figure 3 presents a KE tableau for $p \vee q, p \vee \neg q, \neg p \vee q, \neg p \vee \neg q \vdash r$.

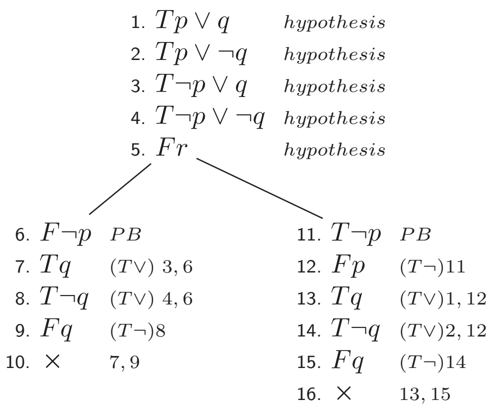


FIG. 3. A KE-Tableau Deduction

3 Branch-driven Abduction

We start our presentation with an example. Suppose we have the following sequent, for which we want to add an extra hypothesis H that makes it provable

$$p \to r, p \vee q \vdash^? r.$$

First we try and build a KE-proof for it. Typically we run the proof-procedure in Algorithm 2.1 until it normally halts with a completed KE-tableau, and then we generate the abduced hypothesis. Alternatively, we could decide to stop the procedure in the middle and perform abduction on the resulting uncompleted KE-tableau. This decision may depend on several reasons: we have run out of computational resources or — as we shall more explicitly suggest later on — such an "on-the-fly" abduction may be part of our theorem-proving strategy.

In this simple example, however, the procedure halts normally, leading to the following completed (one-branch) KE-tableau.

1.	$Tp \to r$	*hypothesis*
2.	$Tp \vee q$	*hypothesis*
3.	Fr	*hypothesis*
4.	Fp	$(T\to_2)$ 1,3
5.	Tq	$(T\vee_1)$ 2,4

At this point, one collects all the unfulfilled signed formulas, which occur in the only branch of this KE-tableau. Since there are no unfulfilled non-atomic signed formulas, our collection contains only the set of signed atomic formulas $\Psi = \{Fr, Fp, Tq\}$.

Each nonempty subset $\Phi \subseteq \Psi$ generates a possible candidate for the sought-after extra hypothesis. This is, essentially, a formula H such that its truth is easily seen to be inconsistent with the information contained in Φ. Proceeding with our example, the following candidates can be selected:

Candidate	Φ	H
1	$\{Fp\}$	p
2	$\{Tq\}$	$\neg q$
3	$\{Fr\}$	r
4	$\{Fp, Tq\}$	$q \to p$
5	$\{Fp, Fr\}$	$p \vee r$
6	$\{Tq, Fr\}$	$q \to r$
7	$\{Tq, Fp, Fr\}$	$q \to (p \vee r)$

The only candidate that is rejected is number 3, on the grounds that it violates the condition $H \nvdash G$, that is, the goal must not be provable from the abduced hypothesis on its own, which fails for candidate 3 as $r \vdash r$ is provable.

If we use the set of *all* the unfulfilled hypotheses in the branch, that is the whole of Ψ, we obtain *the least compromising hypothesis* (or LCH for short), in the sense that all other possible hypotheses imply it, but it does not imply any of the others.

3.1 The Abductive Procedure

The generic computation of a hypothesis from a set of signed formulas is as follows. Given a nonempty set of signed formulas $\Phi = \{TA_1, \ldots, TA_n, FB_1, \ldots, FB_m\}$, we compute the hypothesis $H(\Phi)$ as follows:

$$H(\Phi) = \begin{cases} \neg A_1 & , \textit{if } n=1, m=0 \\ B_1 & , \textit{if } n=0, m=1 \\ \neg(A_1 \wedge \ldots \wedge A_n) & , \textit{if } n>1, m=0 \\ B_1 \vee \ldots \vee B_m & , \textit{if } n=0, m>1 \\ (A_1 \wedge \ldots \wedge A_n) \rightarrow (B_1 \vee \ldots \vee B_m), & \textit{if } n>0, m>0 \end{cases}$$

Lemma 3.1 *Given a KE-tableau branch containing formulas Φ, if we add $H(\Phi)$ as a top hypothesis, then this branch can be expanded into a closed subtree.*

Proof By simple application of KE rules. In the most general case, we have a PB application over $A_1 \wedge \ldots \wedge A_n$ at the point the branch expansion stopped, as shown below.

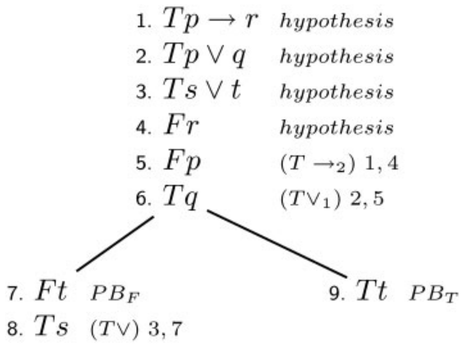
$$\begin{tikzcd} & \begin{array}{c} T(A_1 \wedge \ldots \wedge A_n) \rightarrow (B_1 \vee \ldots \vee B_m) \\ TA_1 \\ \vdots \\ TA_n \\ FB_1 \\ \vdots \\ FB_m \end{array} \arrow[dl, dash] \arrow[dr, dash] & \\ \begin{array}{l} FA_1 \wedge A_2 \wedge \ldots \wedge A_n \\ FA_2 \wedge \ldots \wedge A_n \\ \quad \vdots \\ FA_n \\ \ \times \end{array} & & \begin{array}{l} TA_1 \wedge A_2 \wedge \ldots \wedge A_n \\ TB_1 \vee \ldots \vee B_m \\ \quad \vdots \\ TB_m \\ \ \times \end{array} \end{tikzcd}$$

So both branches of the KE-tableau close. ∎

If the initial tableau has more than one open branch, then we have to apply this method to each branch. For example, suppose we alter the initial sequent of the previous example to

$$p \rightarrow r, p \vee q, s \vee t \vdash^{?} r$$

so that the initial tableau expansion is now



By applying the ideas described so far, the left branch leads to a hypothesis $H_l = (q \wedge s) \to (p \vee r \vee t)$, while the right branch leads to the hypothesis $H_r = (q \wedge t) \to (p \vee r)$; note that we have abduced the least compromising hypotheses, using the complete set of unfulfilled signed formulas in hand. As a result, if we add the conjunction of these hypotheses $H_l \wedge H_r$ to the original sequent, the KE-tableau will certainly close. So, when there are more than one branch, it suffices to take the conjunction of all abduced hypotheses.

Note that if the abduction procedure had been applied prior to the branching, also aiming at the less compromising formula, the signed formula $Ts \vee t$ would have been unfulfilled and the abduced hypothesis would have been $H = (q \wedge (t \vee s)) \to (p \vee r)$. It turns out that H and $H_l \wedge H_r$ are logically equivalent; however, H is smaller and provides a shorter proof. From that we learn that, if both sub-branches after a PB application remain open, it may be better to apply the abduction procedure prior to the branching. We shall come back to this point in Section 5 below.

Algorithm 3.1 Branch-driven Abduction

```
BranchAbduction($\Gamma, G, \mathcal{T}$)
Input: A sequent $\Gamma \vdash^? G$, and $\mathcal{T}$ a partially expanded tableau for it
Output: A hypothesis $H$ such that $\Gamma, H \vdash G$

1: Let $\mathcal{B}_1, \ldots, \mathcal{B}_k$ be the open branches in $\mathcal{T}$.
2: for $i = 1$ to $k$ do
3:     Let $\Psi_i = \{XA \in \mathcal{B}_i | XA$ unfulfilled in $\mathcal{B}_i\}$
4:     Choose $\Phi_i \subseteq \Psi_i$
5:     Let $H_i = H(\Phi_i)$
6: end for
7: return $H_1 \wedge \ldots \wedge H_k$
```

Algorithm 3.1 presents the branch-driven abduction algorithm. Note that it is a non-deterministic algorithm in a twofold way. First, it lets one expand the tableau in whatever fashion one wants, and the halting of this expansion is not specified; second, one can choose the subset Φ of unfulfilled signed formulas to generate the abduced hypothesis.

Theorem 3.1 (Correctness of Algorithm 3.1) *Algorithm 3.1 is correct, that is, on input $\Gamma \vdash^? G$ it outputs a formula H such that $\Gamma, H \vdash G$.*

Proof Let $H = H_1 \wedge \ldots \wedge H_k$ be an answer computed by Algorithm 3.1. It suffices to note that Lemma 3.1 guarantees that each H_i allows us to expand an open branch into a closed subtree, so all branches end up closed. ∎

4 KE-based vs tableau-based abduction

The abductive procedure described in the previous section is not specific to the KE method, but could be equally applied to traditional Smullyan-style tableaux. In this section we shall show, however, how the inefficient behaviour of Smullyan-style tableaux (highlighted in [10, 11, 12,]) heavily affects the output of the abductive procedures, leading to *redundant* abduced hypotheses. The problem lies in the fact that applying the procedure to a Smullyan-style tableau the LCH's associated with distinct branches *are not guaranteed to be logically independent of each other.* As a very simple example, consider the sequent

$$p \vee q, p \vee \neg q \vdash^{?} \bot. \tag{2}$$

It is easy to check that, if branch-driven abduction is performed on the completed Smullyan-style tableau, the abduced hypotheses associated with the two branches of this tableau are $\neg p$ and $\neg(p \wedge q)$. Clearly, the latter hypothesis is redundant, being logically implied by the former.

As the complexity of the tree grows, this kind of redundancy may become a major source of inefficiency, especially when it is not easy to recognise whether the LCH associated with one branch is logically implied by the conjunction of the others.

Another example may help to clarify this point. Consider the sequent

$$\neg(p \vee q) \vee (p \wedge \neg r) \vee (\neg q \wedge \neg r) \vdash^{?} \bot. \tag{3}$$

A completed Smullyan-style tableau for this sequent and the LCH's associated with each branch are shown in *Figure* 4. Here the LCH's associated with the first two branches logically imply the one associated with the third. It can be highly expensive on computational resources to remove this kind of redundancy. In fact, this task involves, in general, solving k distinct deduction problems for a tableau with k branches, each of which can be a non-trivial one.

By way of contrast, we can show that such a redundancy can never arise when the abduction procedure is applied to a KE-tableau. In fact, the LCH associated with one branch of a KE-tableau is *never* logically implied by the conjunction of the others (unless it is a tautology). Figures 5 and 6 show this in connection with examples in (2)and (3). We now prove that this is true in general. Let us denote by $\mathrm{LCH}_{\mathcal{B}}$ the least compromising hypothesis associated with branch $\mathcal{B}$. Let us also write $H(XA)$ as an abbreviation for $H(\{XA\})$, so that $H(TA) = \neg A$ and $H(FA) = A$. Then the above claim is expressed by the following:

Theorem 4.2 *Let $\mathcal{T}$ be a KE-tableau with open branches $\mathcal{B}_1, \ldots, \mathcal{B}_n$. Then, for every i, j such that $i \neq j$, if $\mathrm{LCH}_{\mathcal{B}_j}$ is not a tautology,*[2] *then* $\bigwedge_{i \neq j} \mathrm{LCH}_{\mathcal{B}_i} \nvdash \mathrm{LCH}_{\mathcal{B}_j}$.

[2]Observe that if $\mathrm{LCH}_{\mathcal{B}_j}$ is a tautology, then $\mathcal{B}_j$ can be expanded into a closed subtree of $\mathcal{T}$ and so the LCH associated with it is not really needed in order to prove the sequent for which the KE tableau had been built. Thus, this case can never arise when the abduction procedure is applied to a completed KE-tableau. On the other hand, it can arise when the procedure is applied to a partially expanded KE-tableau.

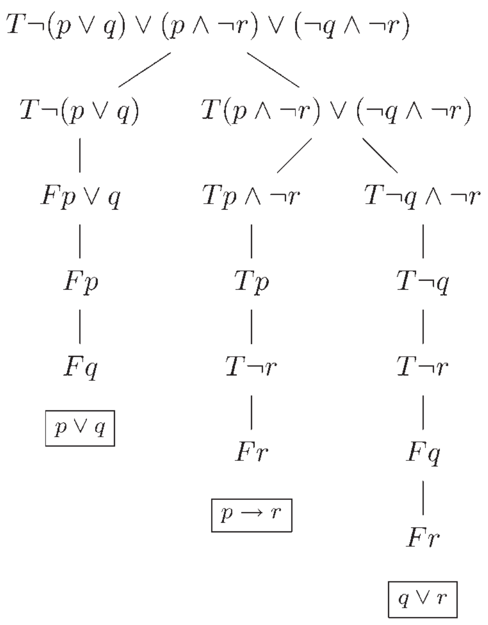


FIG. 4. A completed Smullyan tableau for the sequent in (3). The box underneath each branch shows the least compromising hypothesis associated with it by the abduction procedure.

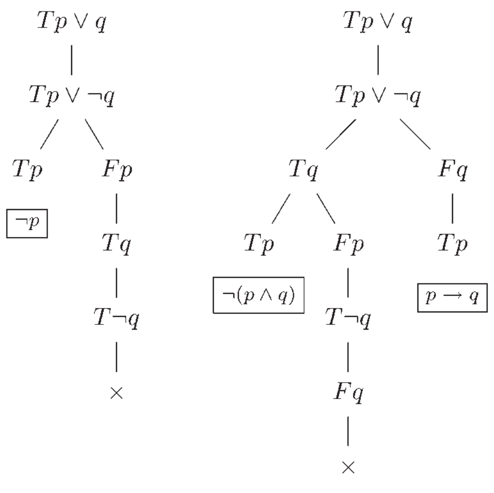


FIG. 5. Two completed KE-tableaux for the sequent in (2).

To show the theorem, we first prove the following:

Lemma 4.1 If $XA \in \mathcal{B}$, then $H(XA) \vdash \mathrm{LCH}_{\mathcal{B}}$.

Proof To show the lemma, consider a branch $\mathcal{B}$ and a signed formula XA occurring in it. The proof is by induction on the logical complexity of XA, i.e. the number of occurrences of logical operators contained in it.

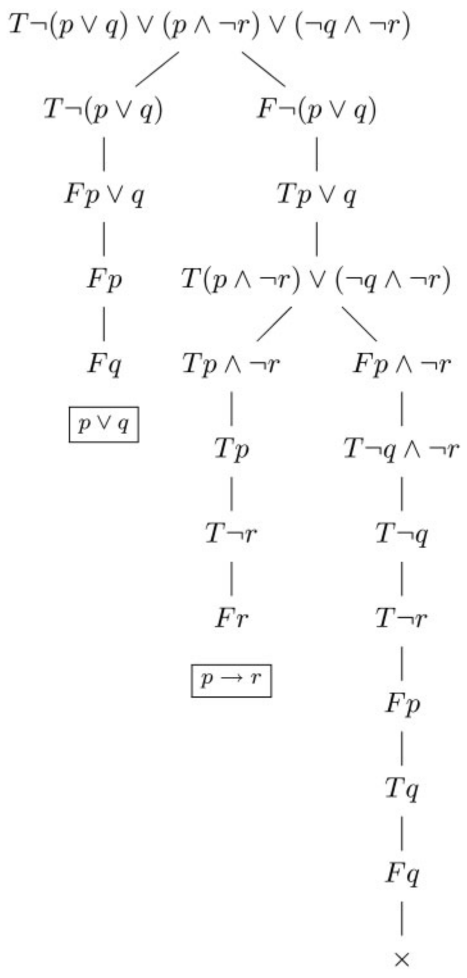


FIG. 6. A completed KE-tableau for the sequent in (3).

Base: XA is an atomic signed formula, that is, $A = p$ for some atomic p. Then XA is, by definition, unfulfilled and belongs to the set Ψ defined in Step 3 of Algorithm 3.1. Since the LCH for $\mathcal{B}$ is obtained by choosing $\Phi = \Psi$ in the next step, $\mathrm{LCH}_{\mathcal{B}}$ will have the form $p \wedge B \to C$ if $X = T$ and $B \to C \vee p$ if $X = F$. In the first case, $H(XA) = \neg p$, while in the second $H(XA) = p$. Hence, in either case $H(XA) \vdash \mathrm{LCH}_{\mathcal{B}}$.

Step: The logical complexity of XA is greater than 0. If XA is unfulfilled in $\mathcal{B}$, then the thesis follows by the same argument used for the base case. If XA if fulfilled in $\mathcal{B}$ we distinguish two cases.

First, suppose XA is an α (see Table 1). From the assumption that XA is fulfilled, it follows that both α_1 and α_2 are in $\mathcal{B}$. Since the complexity of both α_1 and α_2 is strictly smaller than the complexity of α, by inductive hypothesis we can conclude that $H(\alpha_1) \vdash \mathrm{LCH}_{\mathcal{B}}$ and

$H(\alpha_2)\vdash \mathrm{LCH}_B$, and therefore $H(\alpha_1)\vee H(\alpha_2)\vdash \mathrm{LCH}_B$. Now, it is not difficult to verify that, for every α, $H(\alpha)\equiv H(\alpha_1)\vee H(\alpha_2)$ and so $H(\alpha)\vdash \mathrm{LCH}_{\mathcal{B}}$.

Second, suppose XA is a β. From the assumption that XA is fulfilled, it follows that either β_1 or β_2 is in $\mathcal{B}$. By inductive hypothesis we can conclude that either $H(\beta_1)\vdash \mathrm{LCH}_B$ or $H(\beta_2)\vdash \mathrm{LCH}_B$ and therefore $H(\beta_1)\wedge H(\beta_2)\vdash \mathrm{LCH}_B$. Now, it is not difficult to verify that, for every β, $H(\beta)\equiv H(\beta_1)\wedge H(\beta_2)$ and so $H(\beta)\vdash \mathrm{LCH}_{\mathcal{B}}$. ∎

Given Lemma 4.1, Theorem 4.1 is proved by the following argument.

Suppose for some j such that $\mathrm{LCH}_{\mathcal{B}_j}$ is not a tautology,

$$\bigwedge_{i\neq j}\mathrm{LCH}_{\mathcal{B}_i}\vdash \mathrm{LCH}_{\mathcal{B}_j}. \tag{4}$$

Consider an arbitrary branch $\mathcal{B}_k$ with $k\neq j$. By definition of KE-tableau, given two distinct branches $\mathcal{B}_k$ and $\mathcal{B}_j$ there must be signed formulas XA and $\bar{X}A$ such that XA occurs in $\mathcal{B}_k$ and $\bar{X}A$ occurs in $\mathcal{B}_j$ (that is an application of the PB rule which originates the branching). So, by Lemma 4.1,

$$H(XA)\vdash \mathrm{LCH}_{\mathcal{B}_k} \tag{5}$$

$$H(\bar{X}A)\vdash \mathrm{LCH}_{\mathcal{B}_j}. \tag{6}$$

Now, by the assumption that $\bigwedge_{i\neq j}\mathrm{LCH}_{\mathcal{B}_i}\vdash \mathrm{LCH}_{\mathcal{B}_j}$ and cut, we can conclude from (5), that $\bigwedge_{i\neq j,k}\mathrm{LCH}_{\mathcal{B}_i}, H(XA)\vdash \mathrm{LCH}_{\mathcal{B}_j}$ and so, given (6):

$$\bigwedge_{i\neq j,k}\mathrm{LCH}_{\mathcal{B}_i}, H(XA)\vdash \mathrm{LCH}_{\mathcal{B}_j} \qquad \text{and} \qquad H(\bar{X}A)\vdash \mathrm{LCH}_{\mathcal{B}_j}.$$

However, $H(XA)$ and $H(\bar{X}A)$ are complementary formulas (i.e. one is the negation of the other), and therefore $\bigwedge_{i\neq j,k}\mathrm{LCH}_{\mathcal{B}_j}$, by the propositional calculus. By repeating the same argument for all the branches $\mathcal{B}_i$ with $i\neq j$ we can eliminate, one by one, all the associated LCH's from the antecedent of the sequent in (4) and conclude that $\mathrm{LCH}_{\mathcal{B}_j}$ must be a tautology, against one of the assumptions. This concludes the proof of Theorem 4.1.

The logical independence of the LCH's associated with one branch from the conjunction of the others is a consequence of a crucial feature of KE-tableaux which, in turn, descends from their use of the principle of bivalence as the *only* branching rule: given any two distinct branches $\mathcal{B}_i$ and $\mathcal{B}_j$ there exist conjugate signed formulas XA and $\bar{X}A$ such that XA is in $\mathcal{B}_i$ and $\bar{X}A$ is in $\mathcal{B}_j$. Other refutation systems with a different "branching policy", such as Smullyan's tableaux, may well generate trees where the LCH's associated with two distinct branches are not logically independent, so that their conjunction (the abduced hypothesis) is redundant.

5 Dynamic Abduction

In the examples viewed so far, the input sequents are not provable, so that these examples belong to the traditional "explanationist" view of abduction. On the other hand, the abduction Algorithm 3.1 does not state at which point in the deduction it should be applied. This allows for the distinction between post-hoc (or static) abduction, which applies to a completed open KE-tableau, and on-the-fly (dynamic) abduction, which applies to an uncompleted, that is, partially expanded KE-tableau.

In the latter case, it is not known in advance whether the abduced formula generated by Algorithm 3.1 is a genuine extra hypothesis, in which case the sequent is not provable, or it is a “lemma”, that is, it is itself a theorem which can be used as an intermediate step, in which case the sequent is indeed provable. This way of interpreting abduction is in fact closer to real scientific practice, in which “lemmas” play a prominent role in simplifying proofs, either by capitalizing on previously established knowledge or by reducing a complex problem to one which is easier to solve (lemmas are often stated first as extra hypotheses and are “proven away” only at a later stage). Sticking to the domain of classical propositional logic, to which the present paper is restricted, the abduced lemmas may be tautologies that have already been proven or that can be easily proved from a given logical repertoire. Moreover, and more importantly, it would be highly desirable to present an “on-the-fly” abduction procedure that is able, whenever the sequent is provable but no short analytic proofs exist, to generate non-analytic proofs which are shorter than the shortest analytic ones.

For this form of dynamic abduction, one needs an *abduction strategy* of *abduction heuristics*, that allows one to choose the abduction parameters. Such abduction heuristics has to decide:

- When to apply abduction?
- Which formula do abduce?

Each such heuristics will be analysed as to whether it leads to a complete proof method and if the decision procedure is terminating.

5.1 Naive Dynamic Abduction

Let us now consider an example that is provable, but belongs to a family of hard theorems known as the Tseitin formulas [23].

Example 5.1 Consider the Tseitin sequent[3],

$$\begin{array}{c} (p\wedge\neg q)\vee(\neg p\wedge q), \\ (q\wedge\neg r)\vee(\neg q\wedge r), \\ (p\wedge\neg s)\vee(\neg p\wedge s) \end{array} \quad \vdash \quad (r\wedge\neg s)\vee(\neg r\wedge s).$$

[3]A *Tseitin sequent* $\Gamma\vdash\Delta$ is constructed as follows. Consider any undirected graph, in which every edge is associated to a distinct atom, and every node is associated to a formula, namely the exclusive-or of all adjacent edges. An odd number of nodes receive marks. Let Γ be the set of formulas associated to marked nodes, and let Δ be the set of formulas associated to unmarked nodes. Such a sequent is always provable, a fact associated to the property that the sum of degrees of all nodes in a graph is even. Each propositional valuation represents a subgraph, to which the property also applies.

The example in question is obtained by a square graph with edges associated to p, q, r and s, whose only unmarked node is adjacent to edges associated to r and s.

for which one constructs a partially expanded KE-tableau in which only the leftmost branch is closed, so that there are two open branches.

$$\begin{array}{lll}
1. & T(p \wedge \neg q) \vee (\neg p \wedge q) & \textit{hypothesis} \\
2. & T(q \wedge \neg r) \vee (\neg q \wedge r) & \textit{hypothesis} \\
3. & T(p \wedge \neg s) \vee (\neg p \wedge s) & \textit{hypothesis} \\
4. & F(r \wedge \neg s) \vee (\neg r \wedge s) & \textit{hypothesis} \\
5. & F(r \wedge \neg s) & (F\vee)\ 4 \\
6. & F(\neg r \wedge s) & (F\vee)\ 4
\end{array}$$

$$\begin{array}{lll}
7. & F(p \wedge \neg q) & PB_l \\
8. & T(\neg p \wedge q) & (T\vee)\ 1,7 \\
9. & T\neg p & (T\wedge)\ 8 \\
10. & Tq & (T\wedge)\ 8 \\
11. & Fp & (T\neg)\ 9
\end{array}
\qquad
\begin{array}{lll}
23. & T(p \wedge \neg q) & PB_r \\
24. & Tp & (T\wedge)\ 23 \\
25. & T\neg q & (T\wedge)\ 23 \\
26. & Fq & (T\neg)\ 25
\end{array}$$

$$\begin{array}{lll}
12. & F(q \wedge \neg r) & PB_l \\
13. & T(\neg q \wedge r) & (T\vee)\ 2,12 \\
14. & T\neg q & (T\wedge)\ 13 \\
15. & Tr & (T\wedge)\ 13 \\
16. & Fq & (T\neg)\ 14 \\
17. & \times & 10,16
\end{array}
\qquad
\begin{array}{lll}
18. & T(q \wedge \neg r) & PB_r \\
19. & Tq & (T\wedge)\ 18 \\
20. & T\neg r & (T\wedge)\ 18 \\
21. & Fr & (T\neg)\ 20 \\
22. & Fs & (F\wedge)\ 6,20
\end{array}$$

Algorithm 3.1 is then applied to it, selecting only signed atomic formulas at each open branch for the construction of the abduced hypotheses.

The two open branches lead to abduction hypothesis $H_1 = q \to (p \vee r \vee s)$ and $H_2 = p \to q$, so that the global abduced hypothesis is $H_1 \wedge H_2$. This hypothesis was obtained in exactly the same manner as in the cases where the tableau did not close. But here we have that both $\Gamma \vdash G$ and $\Gamma, H_1 \wedge H_2 \vdash G$.

Note that the initial KE-proof procedure was halted at an arbitrary point. That is, the expansion of the KE-tableau was done, using the procedure outlined in Algorithm 2.1, in such a way that some branches were left open without any further attempts to apply the PB rule. It remains to be shown how a full proof of the initial tableau can be achieved.

The formula abduced above, $H_1 \wedge H_2$, is such that $\Gamma, H_1 \wedge H_2 \vdash G$. To obtain a proof of $\Gamma \vdash G$, a proof of $\Gamma \vdash G, H_1 \wedge H_2$ is constructed, and with an application of the cut rule over the abduced formula, the desired result follows.

Is this more efficient than trying to prove $\Gamma \vdash G$ directly? Unfortunately not. The abduced hypothesis is constructed in such a way that, if we construct an analytic KE-proof for $\Gamma \vdash G, H_1 \wedge H_2$, in exactly the same manner as we built a proof in the abduction of $H_1 \wedge H_2$, and such that initially we branch on H_2, we end up with exactly the same open branches containing the same unfulfilled formulas as we had when the proof was stopped during the abduction process. This is shown below, where this partly expanded a KE proof of $\Gamma \vdash G, H_1 \wedge H_2$ is presented.

1.	$T(p \wedge \neg q) \vee (\neg p \wedge q)$	*hypothesis*
2.	$T(q \wedge \neg r) \vee (\neg q \wedge r)$	*hypothesis*
3.	$T(p \wedge \neg s) \vee (\neg p \wedge s)$	*hypothesis*
4.	$F(r \wedge \neg s) \vee (\neg r \wedge s)$	*hypothesis*
5.	$F[q \rightarrow (p \vee r \vee s)] \wedge [p \rightarrow q]$	*abduced*
6.	$F(r \wedge \neg s)$	$(F\vee)$ 4
7.	$F(\neg r \wedge s)$	$(F\vee)$ 4

8.	$Tp \rightarrow q$	PB_l
9.	$Fq \rightarrow (p \vee r \vee s)$	$(F\wedge)$ 5, 8
10.	Tq	$(F \rightarrow)$ 9
11.	$Fp \vee r \vee s$	$(F \rightarrow)$ 9
12.	Fp	$(F \rightarrow)$ 11
13.	Fr	$(F \rightarrow)$ 11
14.	Fs	$(F \rightarrow)$ 11

15.	$Fp \rightarrow q$	PB_l
16.	Tp	$(F \rightarrow)$ 15
17.	Fq	$(F \rightarrow)$ 15

Note that each branch has, at best, the same unfulfilled signed formulas as the respective branches used to compute the abduced hypothesis. Also note that the abduced formula is larger than the other formulas in the hypothesis, and it contains all atomic symbols in the sequent, while the other formulas are concerned with a restricted number of symbols.

With regards to a dynamic abduction heuristics, the naive method can be described by:

- When to apply abduction?
  After applying all linear expansion rules
- Which formula do abduce?
  LCH.

As seen above, this form of heuristics is non-terminating, for the tableau has the same unfulfilled formulas after applying abduction as it had before.

This means that abduction is not a panacea. It can give extra hypotheses that make the original proof more efficient, but a naive approach will not guarantee a more efficient proof for the input sequent, $\Gamma \vdash G$.

5.2 Abducing Proof Efficiency via Dynamic Abduction

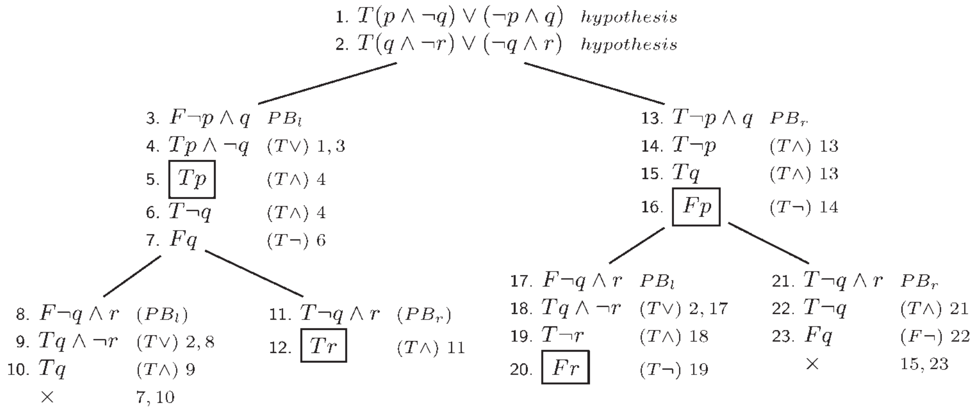
We now show that computing shorter non-analytic proofs is in fact possible for the Tseitin sequent presented in Example 5.1 if a more sophisticated approach is adopted. This time, instead of building a normal tableau, several subsets of the signed formulas will be chosen after an initial linear saturation of the tableau. Each subset will not generate in general a closed tableau; if one subset generates a closed tableau, the search space shrinks. So the abduction procedure is applied to the completed tableau generated by a subset of the original formulas.

The following will be used as guidelines of this process:

- The formulas in a selected subset must have some atoms in common.
- The abduction process will *not* apply the least compromising selection of formulas. The reason is two-fold. First, Example 5.1 showed that not much is gained with such choice;

second, the aim is to eliminate from the abduced formulas some or all of the common atoms, so as to promote a reduction in the "dimension" of the problem. This can be done safely whenever such atom occurs only in the selected subset of formulas, and nowhere else in that branch.

Example 5.2 In the main branch of the tableau for the Tseitin sequent in Example 5.1, we chose the formulas containing the atom q, namely the first two formulas, and develop a complete analytic KE-tableau for it. The analycity is guaranteed by choosing an unfulfilled formula on a linearly saturated branch and applying PB over one of its immediate subformulas (this is sometimes called the *branching heuristics* for analytic KE).


1. $T(p \wedge \neg q) \vee (\neg p \wedge q)$ *hypothesis*
2. $T(q \wedge \neg r) \vee (\neg q \wedge r)$ *hypothesis*
3. $F\neg p \wedge q$ PB_l
4. $Tp \wedge \neg q$ $(T\vee)$ 1, 3
5. $\boxed{Tp}$ $(T\wedge)$ 4
6. $T\neg q$ $(T\wedge)$ 4
7. Fq $(T\neg)$ 6
8. $F\neg q \wedge r$ (PB_l)
9. $Tq \wedge \neg r$ $(T\vee)$ 2, 8
10. Tq $(T\wedge)$ 9
$\times$ 7, 10
11. $T\neg q \wedge r$ (PB_r)
12. $\boxed{Tr}$ $(T\wedge)$ 11
13. $T\neg p \wedge q$ PB_r
14. $T\neg p$ $(T\wedge)$ 13
15. Tq $(T\wedge)$ 13
16. $\boxed{Fp}$ $(T\neg)$ 14
17. $F\neg q \wedge r$ PB_l
18. $Tq \wedge \neg r$ $(T\vee)$ 2, 17
19. $T\neg r$ $(T\wedge)$ 18
20. $\boxed{Fr}$ $(T\neg)$ 19
21. $T\neg q \wedge r$ PB_r
22. $T\neg q$ $(T\wedge)$ 21
23. Fq $(F\neg)$ 22
$\times$ 15, 23

The two middle branches are open. On each open branch, the formulas selected for constructing the branch abduction are framed in boxes. Note that no formula containing q is selected. Over such selection, the abduction algorithm yields

$$\neg(p \wedge r) \wedge (p \vee r)$$

The fact that this formula is of the same size as the input formulas is an indication that the "dimension reduction" is applicable here, that is, that the method provided a reduction in the number of variables without causing an exponential explosion on the size of formulas.

The example proceeds with a second abduction on the formulas involving the atom s.

Example 5.3 The construction of a "small" proof for the Tseitin sequent proceeds by considering an analytic KE-tableau built over the formulas containing s, namely,

$$T(p \wedge \neg s) \vee (\neg p \wedge s)$$
$$F(r \wedge \neg s) \vee (\neg r \wedge s)$$

which corresponds to lines 3 and 4 in the tableau of Example 5.1. The completed expansion of this tableau has two closed and two open branches (details omitted), one with atomic formulas $\{Tp, Fr, Fs\}$, and the other containing $\{Fp, Tr, Ts\}$. The abduction procedure, by eliminating s, produces

$$(p \rightarrow r) \wedge (r \rightarrow p)$$

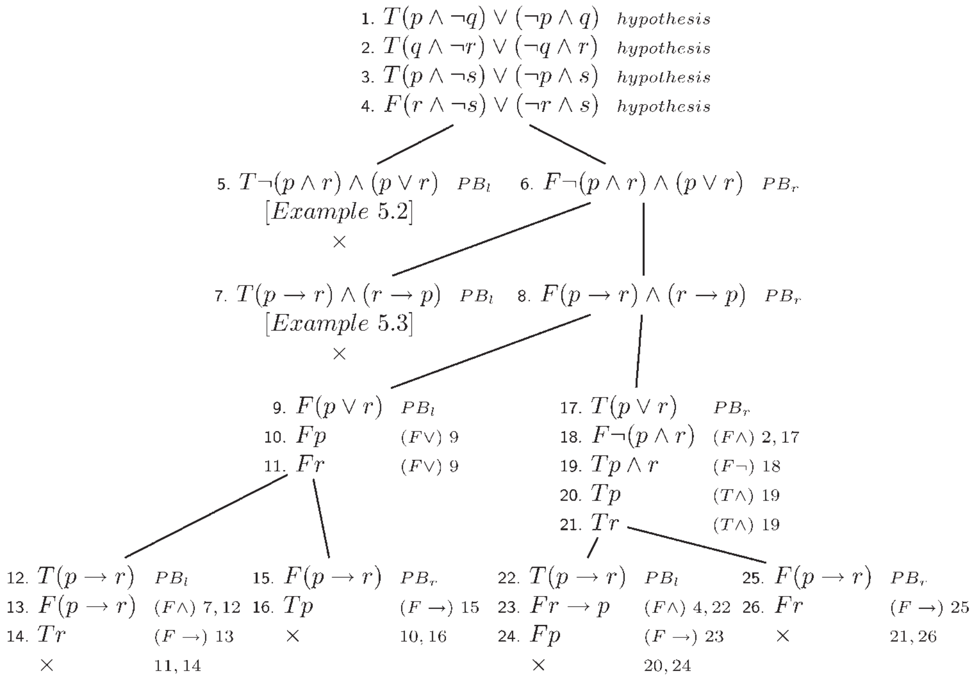
which is also of the same size as the input formulas.

We are now in a position to obtain a full proof the Tseitin sequent, which we claim to be "small".

Example 5.4 Consider the Tseitin sequent,

$$\begin{array}{c} (p\wedge\neg q)\vee(\neg p\wedge q), \\ (q\wedge\neg r)\vee(\neg q\wedge r), \\ (p\wedge\neg s)\vee(\neg p\wedge s) \end{array} \quad \vdash \quad (r\wedge\neg s)\vee(\neg r\wedge s).$$

for which one constructs a non-analytic KE-tableau based on the abduction procedure and a final analytic expansion on the abduced formulas.



1. $T(p\wedge\neg q)\vee(\neg p\wedge q)$ *hypothesis*
2. $T(q\wedge\neg r)\vee(\neg q\wedge r)$ *hypothesis*
3. $T(p\wedge\neg s)\vee(\neg p\wedge s)$ *hypothesis*
4. $F(r\wedge\neg s)\vee(\neg r\wedge s)$ *hypothesis*
5. $T\neg(p\wedge r)\wedge(p\vee r)$ PB_l
[*Example* 5.2]
×
6. $F\neg(p\wedge r)\wedge(p\vee r)$ PB_r
7. $T(p\rightarrow r)\wedge(r\rightarrow p)$ PB_l
[*Example* 5.3]
×
8. $F(p\rightarrow r)\wedge(r\rightarrow p)$ PB_r
9. $F(p\vee r)$ PB_l
10. Fp $(F\vee)$ 9
11. Fr $(F\vee)$ 9
17. $T(p\vee r)$ PB_r
18. $F\neg(p\wedge r)$ $(F\wedge)$ 2, 17
19. $Tp\wedge r$ $(F\neg)$ 18
20. Tp $(T\wedge)$ 19
21. Tr $(T\wedge)$ 19
12. $T(p\rightarrow r)$ PB_l
13. $F(p\rightarrow r)$ $(F\wedge)$ 7, 12
14. Tr $(F\rightarrow)$ 13
× 11, 14
15. $F(p\rightarrow r)$ PB_r
16. Tp $(F\rightarrow)$ 15
× 10, 16
22. $T(p\rightarrow r)$ PB_l
23. $Fr\rightarrow p$ $(F\wedge)$ 4, 22
24. Fp $(F\rightarrow)$ 23
× 20, 24
25. $F(p\rightarrow r)$ PB_r
26. Fr $(F\rightarrow)$ 25
× 21, 26

The closures below nodes 5 and 7 are due to the abduction procedure.

Example 5.4 provides a small proof for the Tseitin sequent because each auxiliary tableau is polynomially bounded and the size of each abduced formula is also polynomially bounded.

With regards to a dynamic abduction heuristics, the *subformula elimination heuristics* can be described by:

- When to apply abduction?
  Choose a subformula A to be eliminated, construct a complete tableau for all formulas containing A.
- Which formula do abduce?
  For each open branch $\mathcal{B}$, compute $H(\Phi)$, where Φ is the set of unfulfilled formulas in $\mathcal{B}$ not containing A.

Clearly, when the eliminated subformula is an atom, the elimination heuristics is terminating, for there are only finitely many atoms to eliminate.

Note that a generic sequence of elimination steps, each of which elimination a set of atoms or subformulas from the sequent, has the potential, at each step, of producing a multiplicative increase in the size of the resulting abduced formula, which can lead to an exponential explosion on the size of the proof.

The example above shows that such explosion does not always occur, but remains a possibility for generic family of sequents. Tseitin sequents have a special structure that makes the application of abduction a good idea. Notably, each atom occurs in exactly two formulas, the number of atom occurrences per formula is uniformly balanced, and there are no "irrelevant" atoms or formulas. So the elimination of atoms tend involve a small number of formulas, and create relatively small tableaux and abduced formulas.

These properties may work as a set of heuristics for the selection of atoms or subformulas for the application of abductive elimination.

The successive atom elimination heuristics is just one example of a method that uses dynamic abduction. There are other possibilities, which may or may not lead to smaller proofs in some cases.

Example 5.5 Another possibility for the application of abductive reasoning is to use the Cut and Pay proof methods [14], which consist of several methods for approximating classical reasoning by limiting the use of the cut rule. In particular, it seems appropriated the abductive use of the parameterised inference $\vdash_k^{CP}$ [4], such that $\Gamma \vdash_k^{CP} \Delta$ if there exists a KE-tableau using at most k *analytic* uses of branching rule PB; that is, at most k applications of analytic cuts.

Clearly if $\Gamma \vdash_k^{CP} \Delta$ then classically $\Gamma \vdash \Delta$. Also, if a tableau for $\Gamma \vdash_k^{CP} \Delta$ has a saturated branch, them classically $\Gamma \nvdash \Delta$. For a fixed k, this inference is decidable in polynomial time in the size of Γ, Δ. Furthermore, $\vdash_k^{CP}$ has the uniform substitution property.

Using this form of approximated reasoning in the abductive steps, there is no limit on the set of premisses used for abductive reasoning. If the auxiliary tableau closes, the sequent has been proved. Otherwise, the dynamic abductive method may be applied.

This limitation on the size of the auxiliary tableaux may be used in conjunction with the selection of abducibles to guarantee that abduced formulas are also of limited size.

However, there is no guarantee that only a polynomial number of applications of this method will yield either a closed tableau or a saturated open branch. In fact, if the combination of Cut and Pay auxiliary deductions and the abduction procedure is not handled with care, this method can lead to the same situation noted in Example 5.1, where the abduction steps only simulate analytic proofs.

This example reinforces that, for a practical use of dynamic abduction, it must be accompanied with a good set of heuristics dependent on the method used for the construction of auxiliary open proofs.

We now formalise the method of dynamic abduction independently of the selection method for auxiliary proofs.

5.3 Dynamic Iterated Abduction Process

Before we describe the dynamic abduction procedure, consider the following result, which is used extensively by it.

[4]This inference was called $\vdash_k^d$ in [14]

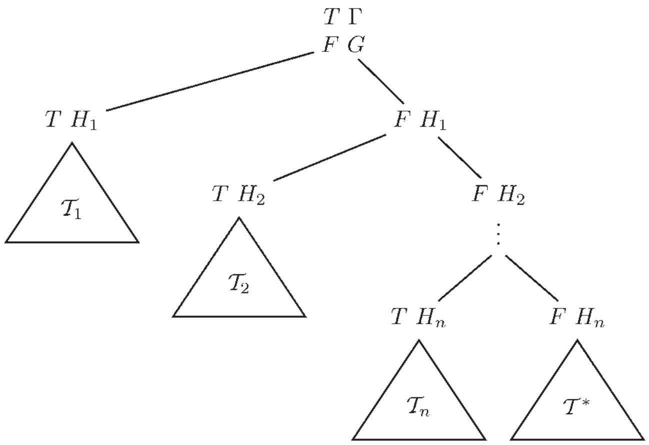
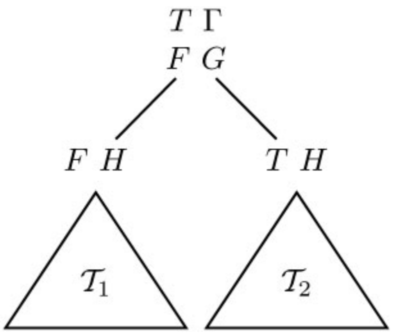


FIG. 7. The Dynamic Abduction Process

Lemma 5.1 *Suppose* $\Gamma, H \vdash G$. *Then* $\Gamma \vdash G$ *iff* $\Gamma \vdash H, G$.

Proof We could show this result using only structural rules from the sequent calculus, but we prove using tableau reasoning instead. We write $T\Gamma$ for $\{TA | A \in \Gamma\}$.

($\Leftarrow$) Assume $\Gamma \vdash H, G$. Then we have a closed tableau for $\Gamma \vdash G$ branching over H



where $\mathcal{T}_1$ is a closed tableau for $\Gamma \vdash H, G$ and $\mathcal{T}_2$ is a closed tableau for $\Gamma, H \vdash G$.

($\Rightarrow$) Assume $\Gamma \vdash G$. Clearly, any closed tableau for $\{T\ \Gamma, F\ G\}$ also closes for $\{T\ \Gamma, FH, FG\}$, so $\Gamma \vdash H, G$ ∎

The dynamic process of repeated applications of the branch abduction algorithm is illustrated in Figure 7.

The idea of the construction on Figure 7 is the following. One starts with the construction of a proof for the original sequent $\Gamma \vdash^? G$. Suppose there is some method for deciding how to start building a finite tableau $\mathcal{T}_1$ for it; it can be either atom elimination, or approximated reasoning, or any decidable method. If $\mathcal{T}_1$, then the proof is finished. Otherwise, the abduction Algorithm 3.1 is applied, yielding an abduced formula H_1. The correctness of the algorithm guarantees that an extension of tableau $\mathcal{T}_1$ for $\Gamma, H_1 \vdash G$ closes. Furthermore, Lemma 5.1 guarantees that the original sequent is provable iff $\Gamma \vdash H_1, G$ is, so the proof proceeds by constructing a tableau for $\Gamma \vdash^? H_1, G$; the KE-method, guarantees there is a proof or refutation for it no longer than a proof or refutation for the original sequent. A proof for the original sequent can be composed with a single application of a potentially non-analytic cut, ie a branching over the abduced formula H_1.

This process can then be iterated, as illustrated in Figure 7. Tableaux $\mathcal{T}_1, \dots, \mathcal{T}_n$ all close due to the correctness of Algorithm 3.1. After every abduction step i, only the rightmost branch, namely the one containing FH_i, is open. At the end of the proof, the rightmost tableau $\mathcal{T}^*$ closes iff the initial sequent is provable, due to iterated applications of Lemma 5.1. When this process terminates we have computed potentially non-analytic cut formulas $H_1, \dots, H_n$, generating a non-analytic proof or refutation for the input sequent.

5.4 The Dynamic Abduction Algorithm

The dynamic abduction algorithm is shown in Algorithm 5.1. The idea is to parameterise it with an abduction heuristic $\mathcal{H}$. Note that the final product is a non-analytic proof for the original sequent.

Algorithm 5.1 Non-analytic Tableau Proof via Dynamic Abduction

```
DynamicAbduction($\Gamma, G, \mathcal{H}$)
Input: A sequent $\Gamma \vdash^? G$ and abduction heuristics $\mathcal{H}$.
Output: A tableau $\mathcal{T}$ for $\Gamma \vdash G$ or a counter-valuation

1: $i := 1$
2: $openBranch := \{TA \mid A \in \Gamma\} \cup \{FG\}$
3: $\mathcal{T} := openBranch$
4: while true do
5:     $\mathcal{T}_i :=$ apply abduction heuristics $\mathcal{H}$ to $openBranch$
6:     if $\mathcal{T}_i$ closes then
7:         Attach $\mathcal{T}_i$ to the end of the open branch in $\mathcal{T}$
8:         return $\mathcal{T}$
9:     else if there is a saturated open branch in $\mathcal{T}_i$ then
10:        return a counter valuation obtained from the open branch
11:    else
12:        $H_i :=$ BranchAbduction($openBranch, \mathcal{T}_i$)
13:        $\mathcal{T}_i :=$ expand and close $\mathcal{T}_i \cup \{TH_i\}$
14:        Expand $\mathcal{T}$ with an application of PB. On the left add $TH_i$ and $\mathcal{T}_i$. On the right add
           $FH_i$
15:        $openBranch := openBranch \cup \{FH_i\}$
16:    end if
17:    $i := i + 1$
18: end while
```

At each iteration step i, a tableaux $\mathcal{T}_i$ is expanded as an application of the abduction heuristics to $\Gamma \vdash^? H_1, \dots, H_{i-1}, G$ *without using the abduced hypothesis* H_i. In fact, H_i can only be computed *after* $\mathcal{T}_i$ is expanded. When $\mathcal{T}_i$ is expanded, there are three possibilities:

- The analytic tableau for $\mathcal{T}_i$ closes (line 6). Then a proof has been constructed for the original sequent $\Gamma \vdash G$.

- The analytic tableau for $\mathcal{T}_i$ has an open saturated branch (line 9). Then a counterexample for $\Gamma \nvdash G$ has been obtained.
- $\mathcal{T}_i$ is open, with no saturated branch. In this case, we can apply Algorithm 12 and compute H_i (line 12). By the correctness of the procedure, $\mathcal{T}_i$ can be closed adding H_i (line 13), so $\mathcal{T}_i$ is expanded an analytic closed tableau for $\Gamma, H_i \vdash H_1, \ldots, H_{i-1}, G$.

The two initial cases are the termination cases of the non-analytic proof. In the last case, the construction of the proof can continue, such that $\mathcal{T}_i$ is a closed sub-tableau. In general, the abduced formula H_i is not a subformula of the original sequent, so the process is very likely to generate a non-analytic proof.

Theorem 5.1 (Partial correctness of Algorithm 5.1) *If Algorithm 5.1 stops, then either it produces a proof of* $\Gamma \vdash G$ *or it produces a counter-valuation for it.*

Proof Suppose Algorithm 5.1 stops after i iterations, such that $\mathcal{T}$ was expanded i times. At each such expansion we can apply Lemma 5.1, so that $\Gamma \vdash G$ iff there is a way to close the right openBranch. The algorithm returns in one of two cases.

In line 8, openBranch has closed and $\mathcal{T}$ is a closed tableau for $\Gamma \vdash G$. In line 5.1, there is a saturated open branch for openBranch from which a counter-valuation for $\Gamma \nvdash G$ can be obtained. ∎

Termination depends on the abduction heuristics chosen. For example, atom elimination is guaranteed to terminate when all atoms have been eliminated.

If the abduction heuristics has the potential of generating infinite proofs, this process can always be interrupted. In this case, a fixed number of iterations k may be also given as part of the heuristics, such that the abduction procedure can be replaced by a simple analytic KE tableau expansion for $\Gamma \vdash^{?} H_1, \ldots, H_k, G$. This last expansion generates the tableau $\mathcal{T}^*$ is Figure 7 and it is guaranteed to always terminate.

6 Conclusion

In this paper, two new methods were presented for the computation of abduced formulas in classical propositional logic. Both methods have in common the fact that it is not required to be known, a priori, if the input sequent is invalid; in case it was valid, the abduced formula provides extra hypothesis that creates a short proof for an expanded problem.

The branch abduction method shows that one does not require an analytic proof method, as suggested by Cialdea-Mayer and Pirri [6, 8], to perform abduction. On the contrary, the fact that no a priori knowledge is required in what concerns the status of the entailment of the goal data from the background theory has shown that abduction can be used as a tool to compute non-analytic proofs, that are potentially smaller than analytic ones.

The abduction-based computation of non-analytic proofs works as a general proof restart. That is, at certain point in the proof development, that proof search is interrupted, closed, and restarted. To avoid wasting all the effort done so far, the abduced formula can be seen as encoding the "lessons learned" in the initial tableau expansion. By using KE tableaux, those lessons are encoded in a compact form, namely that of a conjunction of logically independent formulas. Hopefully, those lessons will be used to find a smaller proof.

There are several possible future works. First, it is now possible to investigate the extension of those results to first-order and other logics. It should be possible to expand the branch-driven abduction method to the first order formulation found in Cialdea-Mayer and

Pirri [6]. Another interesting result in that line, which certainly needs greater care, is to apply the branch-driven abduction ideas to modern tableau-based theorem provers, such as the disconnection calculus [17].

Second, deeper investigation is needed for the computation of compact, non-analytic proofs. We only showed here that such computation is possible. One has still to develop a method that is shown to be efficient and desirable in, at least, some specific cases. Possibly such method will have to combine many other developments in theorem proving apart from the abduction-based restart presented here. The existence of such a method has strict relation with complexity theory, for if a compact proof was shown to always exist, this would imply that NP=co-NP.

References

[1] Atocha Aliseda-Llera. *Seeking explanations: abduction in logic, philosophy of science and artificial intelligence*. PhD thesis, Stanford University, Stanford, CA, USA, 1997.

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

[3] Samuel R. Buss. Some remarks on lengths of propositional proofs. *Archive for Mathematic Logic*, 34:377–394, 1995.

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

[5] Walter Carnielli. Surviving abduction. *Logic Journal of IGPL*, 14(2):237–256, 2006.

[6] Marta Cialdea Mayer and Fiora Pirri. First-order abduction via tableau and sequent calculi. *Journal of the IGPL*, 1(1):99–117, 1993.

[7] Marta Cialdea Mayer and Fiora Pirri. Modal propositional abduction. *Journal of the IGPL*, 3(6):99–117, 1995.

[8] Marta Cialdea Mayer and Fiora Pirri. Abduction is not deduction-in-reverse. *Journal of the IGPL*, 41(1):95–108, 1996.

[9] Simona Colucci, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello. A uniform tableaux-based approach to concept abduction and contraction in aln. In *Description Logics*, 2004.

[10] 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.

[11] 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.

[12] 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.

[13] Robert Demolombe and Luis Fariñas del Cerro. An inference rule for hypothesis generation. In *Proceedings of IJCAI*, pages 152–157, 1991.

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

[15] Dov Gabbay and John Woods. *The Reach of Abduction: Insight and Tria*, volume A Practical Logic of Cognitive Systems, volume 2. Elsevier, 2005.

[16] Dov Gabbay and John Woods. Advice on abductive logic. *Logic Journal of the IGPL*, 14(2):189–219(31), March 2006.

[17] Reinhold Letz and Gernot Stenz. The disconnection tableau calculus. *Journal of Automated Reasoning*, 38(1-3):79–126, 2007.

[18] Charles Sanders Peirce. *Collected Papers of Charles Sanders Peirce*. Harvard University Press, 1931–1958. Volumes 1–8, edited by Charles Hartshorne, Paul Weiss and Arthur Burks.

[19] David Poole. Representing knowledge for logic-based diagnosis. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 1282–1290, 1988.

[20] Raymond Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In *AAAI*, pages 183–189, 1987.

[21] Murray Shanahan. Prediction is deduction but explanation is abduction. In *Proceedings of IJCAI*, pages 1055–1060, 1989.

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

[23] G. S. Tseitin. On the complexity of derivations in the propositional calculus. In A. O. Slisenko, editor, *Structures in Constructive Mathematics and Mathematical Logic, Part II*, pages 115–125. Consultants Bureau, New-York-London, 1968. Translated from Russian.

Received 8 September 2008